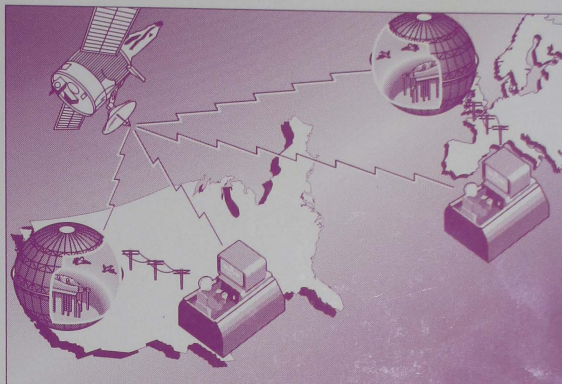


AIAA FLIGHT SIMULATION TECHNOLOGIES CONFERENCE AND EXHIBIT



A COLLECTION OF TECHNICAL PAPERS
DAYTON, OHIO
SEPTEMBER 17 - 19, 1990



For permission to copy or republish, contact:
The American Institute of Aeronautics and Astronautics
370 L'Enfant Promenade, SW
Washington, DC 20024-2518

**A COLLECTION OF
TECHNICAL PAPERS**

**AIAA FLIGHT SIMULATION TECHNOLOGIES
CONFERENCE AND EXHIBIT**

SEPTEMBER 17-19, 1990 / DAYTON, OHIO

**Copyright © by
American Institute of Aeronautics and Astronautics**

All rights reserved. No part of this volume may be reproduced in any form or by any means, electronic or mechanical, including photocopying and recording, without permission in writing from the publisher.

Printed in the U.S.A.

AIAA Flight Simulation Conference
September 17-19, 1990/ Dayton, OH

TABLE OF CONTENTS

Paper No.	Title and Author	Page No.
Session 1- Computer Systems		
90-3140	High Performance Processors for Real-Time Flight Simulation J. Cleveland and D. Crawford and S. Sudik.....	146
90-3141	Techniques for Using Replicated Shared Memory Communications G. Warden.....	W
90-3142	Use of Flight Simulation Accelerators for Enhanced High Speed Computational Capability in Real-Time Manned Flight Simulation P. Moriarty, R. Hollstein, C. Wilsey and J. Wurtz.....	157
90-3143	Advanced Concepts in Real-Time Computing using Virtual Shared Memory D. Baltrus.....	165
Session 2- Crew Station Design		
90-3146	Design, Development and Testing of an Ambient Lighting Simulator for External Illumination of a Transport Simulator Cockpit V. Batson and L. Gupton.....	183
90-3170	Systems and Operational Aspects of Cockpit Automation Insertion and Implementation C. Arndt.....	320
90-3144	5" Size Liquid Crystal Flat Panel Display Evaluation Test by Flight Simulation H. Kawahara, A. Watanabe, K. Wakairo, T. Udagawa, Y. Kurihara and K. Takeda.....	168
90-3145	D Size Liquid Crystal Flat Panel Display Evaluation by Flight Simulator H. Kawahara, K. Wakairo, A. Watanabe, T. Shimizu, F. Morik, K. Matsumura and S. Ohyama.....	176
Session 3- Visual Systems		
90-3147	Environmental Interrogation: Correlating Visual Imagery with the Simulated Mission R. Matusof, B. Hicks and S. Schwain.....	188
90-3149	Real-Time Scene Generator for Testing Optical Seekers W. Wagner.....	195

NA=Not Available
W=Withdrawn

90-3150	Development of In-House TPC Charts to Correlate with the Visual Scene Data Base A. Davoudian.....	W
---------	---	---

Session 4- Modeling

90-3131	The Aerodynamic Effect of Heavy Rain on Airplane Performance D. Vicroy.....	78
90-3132	Adverse Weather Simulation Concepts for Safety of Flight Training B. Montag.....	86
90-3126	Representation of Large-Scale Random Atmosphere Variations for Aerospace Vehicle Simulations K. Dutton, E. Queen, D. Moerder and W. Suit.....	NA
90-3171	The Ship Environment Simulation Problem R. Galloway, F. Frey and D. Carico.....	326

Session 5- Motion Cuing

90-3133	Adaptive Simulator Motion Software with Supervisory Control M. Nahon and L. Reid.....	97
90-3134	Microprocessor Control for a Simulator Motion Platform R. Levi, R. Walker, C. Wilson and L. Hayashigawa.....	106
90-3135	Advanced Techniques for Cuing the Force and Motion Environment in Simulators of the Future Y. Brown, F. Cardullo and G. McMillan.....	115
90-3172	The Effects of Simulator Visual-Motion Asynchrony on Simulator Induced Sickness M. McCauley, L. Hettinger and J. Sinacori.....	NA

Session 6- Simulation Networking

90-3136	Simulation Networking A. Katz.....	123
90-3137	Navigational Simulation Issues for Large Scale Networks J. Burnett, G. George and S. Knight.....	128
90-3138	Interfacing Low Cost Networked Flight Simulators in a SIMNET Environment J. Cadiz, M. Loper, R. Ouyang and J. Thompson.....	132
90-3139	Networked Modular Aircrew Simulation Systems S. Swaine.....	138

Session 7- Verification/Validation

90-3120	The Composition of Flight Simulation Models: Flight Testing Vs. DATCOM Techniques J. Mulder and M. Baarspul	1
---------	---	---

90-3121	Hot Bench Simulation of the Active Flexible Wind-Tunnel Model C. Buttrill and J. Houck.....	11
90-3122	Software Reliability for Flight Simulators G. Stark.....	22
90-3123	Validation Testing: The Drive for Product Quality J. Sarnicola.....	27
90-3173	The Facts and Fictions of Commercial Flight Simulator Validation and Verification B. Hampson.....	337
Session 8- Simulator Research		
90-3127	Power Spectral Analysis to Investigate the Effects of Simulator Time Delay on Flight Control Activity M. Middendorf, S. Lusk and J. Whiteley.....	46
90-3128	Dynamic Seat Cuing with Wide Vs. Narrow Field of View Visual Displays G. McMillan, J. Cress and M. Middendorf.....	53
90-3129	Time Delay Compensation Using Peripheral Visual Cues in an Aircraft Simulator S. Lusk, C. Day, J. Whiteley and W. Johnson.....	63
90-3130	When In-Flight Simulation is Necessary P.Reynolds and V. Calspan.....	71
90-3174	On Analyzing Delays in a Flight Simulation Environment R. McFarland and J. Bunnell.....	341
Session 9- Computational Methods		
90-3151	Two Step Methods for Parameter Estimation for Nonlinear Systems W. Zical.....	202
90-3152	An Improved Method of Integrating Equations of Motion M. Fineberg.....	206
90-3153	The Use of Function Generation in Real-Time Simulation of Stiff System R. Howe and K. Lin.....	217
90-3154	Some Methods for Reducing the Effects of Time Delays in Flight Simulation R. Howe.....	225
90-3155	Optimizing the MPX Operating System for Multi-Tasking Real-Time Programs L. Parker.....	233

NA=Not Available
W=Withdrawn

Session 10- Training Systems

90-3156	The Austower Air Traffic Control Tower Visual Simulator L. Lester.....	W
90-3157	Low Cost Aviation Technology Test Bed B. Goldiez, K. Vliano and D. McBride.....	244
90-3124	Ascent/Abort Training in the Shuttle Mission Simulator J. Sims and M. Sterling.....	35
90-3125	Challenges in the 1990s for Astronaut Training Simulators P. Brown and A. Hajare.....	NA

Session 11- Tactical Simulations

90-3159	Evolution of a Real-Time Air Combat Environment L. Emerson and R. Moore.....	252
90-3160	An Interactive Computerized Aircraft Model for Real-Time Tactical Simulations W. Clark.....	259
90-3161	Air Combat Simulation at USAF Flight Dynamics Laboratory D. Goddard, J. Farley, J. Hackett and S. Banks.....	266
90-3175	Visual Adversary Logic for Close-In Air Combat Simulation B. Montag.....	352
90-3176	User Definable Doctrine for Interactive Air Targets R. Ure.....	359

Session 12- Simulation Applications

90-3162	Rockwell's Real-Time Simulator Aids Hypersonic Vehicle/NASP Designs W. Burnett.....	277
90-3163	Implementation of a Secure Multi-Project Laboratory Facility D. Draffin and J. Bacha.....	284
90-3164	Implementation of a Blade Element UH-60 Helicopter Simulation on a Parallel Computer Architecture in Real-Time B. Moxon and J. Green.....	293
90-3165	Simulation Design Support for the F-16 Multi-Role Fighter From FSD Through MSIP Program Development L. Brumback.....	W
90-3177	Evolution of the Tilt Rotor Flight Simulation During Initial Flight Tests of the V-22 Osprey B. Roberts, M. Joglekar and J. Robertson.....	NA

NA=Not Available

W=Withdrawn

Session 13- Data Bases

90-3166	Project 2851:An Industry Assessment D. Eccles.....	NA
90-3167	Selecting a Remote Sensing Support System for Training Simulation & Mission Rehearsal Applications M. Bromley.....	304
90-3168	Project 2851 Data Base Strategies for Simulator Correlation K. Oda, G. Clayton and B. Lufkin.....	311
90-3169	A Rapid Data Base Configuration System using Multi-Spectral Imagery J. Lickenbrock, K. Spuhl and R. Brown.....	316

THE COMPOSITION OF FLIGHT SIMULATION MODELS:
FLIGHT TESTING VERSUS DATCOM TECHNIQUES

M. Baarspul** and J.A. Mulder*
Delft University of Technology
Faculty of Aerospace Engineering
Kluyverweg 1, 2629 HS Delft, The Netherlands

Abstract

Flight simulation models of the Cessna Citation 500 were composed using two different approaches: DATCOM techniques and identification from flight test data. DATCOM techniques result in so-called a priori airplane models: cost effective alternatives to models obtained by flight testing. Using a high accuracy flight test measurement system, an extensive flight test program was carried out to generate a data base for application of model identification techniques. Next, the quality of the DATCOM a priori models was compared to the models derived from flight tests. To this end part of the flighttest program was used to execute flight test maneuvers for model validation. The quality of the models could be assessed by comparing a posteriori response predictions of these maneuvers to the measured responses. The comparison indicates that for this particular aircraft type acceptable models can be composed using DATCOM techniques. In extreme flight conditions, and for take off and landing, however, flight test data are still essential.

1. Introduction

The present trend in flight simulation for conversion and proficiency training is to exploit flight simulators in the operation of commuter and general aviation aircraft. Environmental restrictions as well as economic and flight safety considerations, and the appearance of 'low cost' flight simulators have no doubt stimulated this trend. A problem is that adequate databases from which aerodynamic-, engine-, mass distribution-, flight control system-, and landing gear models can be composed are usually not available to the simulator manufacturer. It is possible then to embark on a flight test program for identification of these models. If the flight test program is carefully designed, this may be a good approach with the important additional advantage that during the flight test program a POM (Proof of Match) database may be generated which is crucial in FAA Phase II/III simulator certification.

A very cost effective alternative to flight testing is using DATCOM techniques for aerodynamic model building. The problem here is the verification, and tendency of simulator acceptance pilots to enforce subjective changes of certain model parameters during evaluation.

The present paper compares the quality of models derived from flight tests to models resulting from application of DATCOM techniques. The comparison is made for the Cessna Citation 500, a light twin engine jet aircraft, see Fig. 8.

2. DATCOM techniques

2.1 Digital DATCOM

In preliminary design studies a priori aerodynamic models are used which depend on neither windtunnel

nor flighttest measurements. Currently, these models are used in 'low cost' flight simulators for commuter and general aviation aircraft. One of the techniques to obtain these a priori models is the USAF Stability and Control Digital DATCOM.¹ The Digital DATCOM computer program calculates static and dynamic stability derivatives, high-lift and control device characteristics and the effects of the airplane engines. The calculations are based on the methods described in the USAF Stability and Control DATCOM,² using the airplane geometry and inertia characteristics as main inputs. The trim option of the program calculates aerodynamic data and control deflections for airplane trim at subsonic speeds, see Fig. 1.

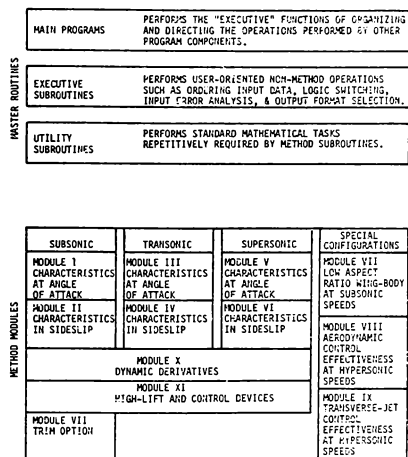


Fig. 1: Digital DATCOM Modules¹

The Digital DATCOM addressable airplane geometries include conventional (including canard), as well as unconventional (multi surface) configurations.

Flight condition definition in Digital DATCOM requires airplane Mach number or velocity, and the atmospheric conditions as either altitude or freestream pressure and temperature, from which the Reynolds number is computed. For a detailed description of Digital DATCOM, reference is made to the User's Manual.¹ In the present work, the program proved to be a valuable tool in the generation of a priori aerodynamic and engine models.

* Professor, Stability and Control

** Associate professor, Stability and Control

2.2 The Smetana program

A second method to compose a priori airplane models is the use of 'Computer Assisted Analysis of Aircraft Performance Stability and Control' by F.O. Smetana.³ This book is written as a detailed user's manual accompanying a package of Fortran computer programs which compute most of the stability and control, and performance characteristics, based on airplane geometry, power plant, and inertial characteristics. However, the Smetana program is restricted to aircraft with moderate-to-high aspect ratio wings, operating at subsonic speeds. Therefore it is suitable in particular to compose general aviation a priori models. The program also includes subroutines which operate on input data specifying the control surface geometry, inertial characteristics, and gearing to compute the control forces and airplane stability and control parameters like neutral points, maneuver points, stick force per g, static stick force stability, maximum attainable roll rate, etc.

2.3 Re-scaling known models

As a third 'method' in the composition of a priori airplane models, it may be possible to make use of available mathematical models and datapackages of similar aircraft. In particular the mass properties model, and the landing gear model,⁴ for which often no data are available, may be subject to re-scaling.

3. A priori airplane models

3.1 General

A priori models for flight simulation should preferably be formulated in accordance with the requirements prescribed by IATA.⁵ The resulting models are rather comprehensive, and include different kinds of non-linearities. Coefficients or functions in these models which cannot be computed with e.g. DATCOM, are set to zero. Some of these coefficients or functions may be identifiable from flight tests later on. As an example, the a priori models developed for the Cessna Citation 500 are briefly described below.

3.2 Aerodynamic model

The aerodynamic model consists of models of the three aerodynamic force coefficients and the three aerodynamic moment coefficients. The model for the aerodynamic lift coefficient, C_L , reads:

$$C_L = C_{L(BASIC)} + C_{L(\delta_e)} + C_{L(\delta_f)} + C_{L(q)} + C_{L(\dot{\alpha})} + C_{L(n)} + C_{L(SB)} + C_{L(UC)} + C_{L(GE)} \quad (1)$$

where C_L denotes the total airplane lift coefficient and $C_{L(BASIC)}$ is the basic airplane lift coefficient, which is developed further into:

$$C_{L(BASIC)} = C_L(\alpha, M) + \Delta C_L(\alpha, \delta_f) + \Delta C_L(M, h) \cdot \alpha \quad (2)$$

Herein $C_L(\alpha, M)$ denotes the lift coefficient of the 'clean configuration' as a function of angle of at-

tack (α) and Mach number (M). It is further expanded into:

$$C_L(\alpha, M) = C_{L_O}(M) + C_{L_\alpha}(M) \cdot \alpha \quad (3)$$

where $C_{L_O}(M)$ is the lift coefficient at α equal to zero, as a function of M , see Fig. 2, and $C_{L_\alpha}(M)$ denotes C_{L_α} as a function of M (Fig. 2A).

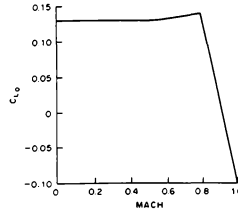


Fig. 2 : C_{L_O} as a function of Mach

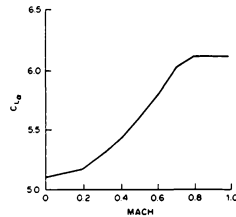


Fig. 2A: C_{L_α} as a function of Mach (clean)

$\Delta C_L(\alpha, \delta_f)$ in (2) denotes the incremental lift as a function of α and wing flap angle (δ_f), see Fig. 3 and $\Delta C_{L_\alpha}(M, h)$ is the increment in C_{L_α} as a function of M and altitude h .

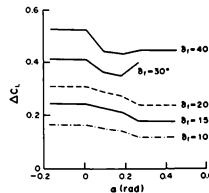


Fig. 3: $\Delta C_L - \alpha$ curves for various flap angles

In (1) $C_{L(\delta_e)}$ denotes the increment in lift coefficient

cient due to elevator deflection (δ_e), which is developed further into:

$$C_{L(\delta_e)} = [C_{L_{\delta_e}}(\alpha, M) + C_{L_{\delta_e}}(\alpha, \delta_f) + \Delta C_{L_{\delta_e}}(M, h) + \Delta C_{L_{\delta_e}}(\delta_e)] \cdot \delta_e \quad (4)$$

$C_{L_{\delta_e}}(\alpha, \delta_f)$ is shown in Fig. 4.

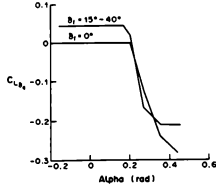


Fig. 4: Increment in $C_{L_{\delta_e}}$ as a function of α and δ_f

$C_{L_{\delta_t}}$ in (1) denotes the increment in lift coefficient, due to elevator trimtab deflection (δ_{te}) and is written as:

$$C_{L_{\delta_t}} = C_{L_{\delta_{te}}}(\alpha, M) \cdot \delta_{te} \quad (5)$$

$C_{L(q)}$ in (1) denotes the increment in lift coefficient due to pitch rate q , and is written as:

$$C_{L(q)} = K_{q_L}(x_{c.g.}) \cdot [C_{L_q}(M, h) + \Delta C_{L_q}(M, \delta_f)] \cdot \frac{\dot{q}}{V} \quad (6)$$

In (6) $K_{q_L}(x_{c.g.})$ denotes the correction factor for the center of gravity (c.g.) location. A similar expression may be written for $C_{L(\dot{\alpha})}$ in (1):

$$C_{L(\dot{\alpha})} = K_{\dot{\alpha}_L}(x_{c.g.}) \cdot [C_{L_{\dot{\alpha}}}(M, h) + \Delta C_{L_{\dot{\alpha}}}(M, \delta_f)] \cdot \frac{\dot{\alpha}}{V} \quad (7)$$

The last four terms in (1) read respectively:

$$C_{L(n)} = [C_{L_n}(M, h) + \Delta C_{L_n}(M, \delta_f)] \cdot (n-1) \quad (8)$$

where n is the normal load factor (g).

$$C_{L(SB)} = K_{SB}(\delta_{SB}) \cdot [C_{L_{SB}}(\alpha, M) + \Delta C_{L_{SB}}(\alpha, \delta_f)] \quad (9)$$

where $K_{SB}(\delta_{SB})$ denotes the speedbrake (SB) effectiveness factor as a function of speedbrake deflection angle (δ_{SB}); $C_{L_{SB}}(\alpha, M)$ and $\Delta C_{L_{SB}}(\alpha, \delta_f)$ are increments in lift coefficient due to full speedbrake deflection as functions of α , M and δ_f

respectively.

$$C_{L(UC)} = K_{UC}(\tau_{LG}) \cdot [C_{L_{UC}}(\alpha, \delta_f) + \Delta C_{L_{UC}}(\alpha, M)](10)$$

where $K_{UC}(\tau_{LG})$ denotes the influence factor for a given degree of undercarriage (UC) extension (τ_{LG}).

$C_{L_{UC}}(\alpha, \delta_f)$ is the change in lift coefficient due to full landing gear extension as a function of α and

δ_f ; $\Delta C_{L_{UC}}(\alpha)$ denotes $\frac{\partial C_{L_{UC}}}{\partial M}$ as a function of α .

$$C_{L(GE)} = K_{GE_L}(h_R) \cdot [C_{L(BASIC)}(\alpha, \delta_f) + C_{L_{\dot{\alpha}_{GE}}}(\delta_f) \cdot \frac{\dot{\alpha}}{V} + C_{L_{\delta_{e_{GE}}}}(\alpha, \delta_f) \cdot \delta_e] \quad (11)$$

where $K_{GE_L}(h_R)$ denotes the groundeffect (GE) influence factor on the lift coefficient as a function of radio altitude (h_R), see Fig. 5.

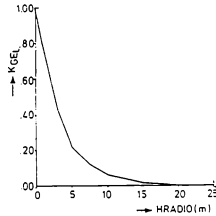


Fig.5: K_{GE_L} as a function of radio altitude

$C_{L(BASIC)}(\alpha, \delta_f)$ is the increment in basic lift coefficient due to ground effect at zero radio altitude as a function of α and δ_f ;

$C_{L_{\dot{\alpha}_{GE}}}(\delta_f)$ is the increment in $C_{L_{\dot{\alpha}}}$ due to groundeffect at zero radio altitude as a function of δ_f ;

$C_{L_{\delta_{e_{GE}}}(\alpha, \delta_f)$ is the increment in $C_{L_{\delta_e}}$ due to groundeffect at zero radio altitude as a function of α and δ_f .

In Ref. 5 similar expressions are given for the aerodynamic drag coefficient (C_D), the sideforce coefficient (C_Y), the pitching moment coefficient (C_m), the rolling moment coefficient (C_l) and the yawing moment coefficient (C_n).

In Section 4.5 the a priori aerodynamic lift coefficient will be compared to the lift coefficient in the final aerodynamic model, resulting from flight test data analysis.

3.3 Engine model

For the a priori engine model, data was obtained from the General Installed Turbofan Performance Computer Program (PI532C) issued by United Aircraft of Canada Ltd. for the commercial version of the Pratt & Whitney (Canada) JT15D turbofan engine. The program, which is of the curve reading type, may be used to generate power setting values of fan speed, resulting in not less than minimum rated thrust on production engines. An example of the calculations performed using this program is presented in Fig. 6, where pressure corrected net thrust XN/δ_1 is plotted versus temperature corrected fan speed $N_1/\sqrt{\theta_2}$ as a function of M and flight level (FL).

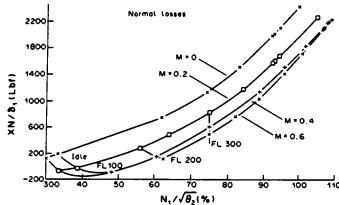


Fig. 6: Net thrust vs fan speed as a function of Mach and Flight Level

The major further development of the engine model was the transient model for realistic variations of all engine parameters with Power Lever Angle (PLA) command, including start-up and shut-down, see Section 4.5.

3.4 Mass properties model

The mass properties model, which describes the centre of gravity (c.g) position and the moments of inertia of the Citation 500 for several loading conditions, was developed using the Weight and Balance Manual of the Citation. The moment and product of inertia envelopes were calculated for several configurations, using calculated fuel inertia and inertia data of the complete aircraft. As an example, the yawing moment of inertia envelope resulting from these calculations, is shown in Fig. 7.

3.5 Flight control system model

The flight control system of the Citation 500 is fully mechanical. Aerodynamic hinge moments, mechanical inertia, friction and flexibility must therefore be included in a model of this flight control system. A second order system model was used for the primary flight control dynamics. The equation of motion, used for the computation of the control column-, control wheel- and rudder pedal deflections (s) reads:

$$F = m\ddot{s} + W\dot{s} + Cs + F_{AER} + F_{COUL} + F_{STOP} + F_G + F_{NOSE}$$

where F denotes the control force applied by the pilot, m is the reduced mass of the flight controls, W denotes the viscous friction and C the spring stiffness. F_{AER} , F_{COUL} , F_{STOP} denote the aerodynamic, coulomb and mechanical stop force

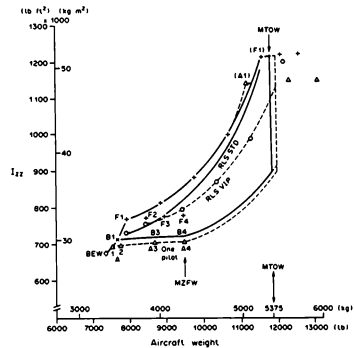


Fig. 7: Yaw moment of inertia envelope

respectively, F_G denotes the force due to static unbalance and F_{NOSE} is the nose-wheel steering force on the rudder pedals, the Citation nose wheel steering being connected to these pedals via springs. In Section 4.5, the final flight control system model will be described which resulted from the identification of the Citation 500 flight controls, using flight test measurements.

3.6 Landing gear model

In the a priori landing gear model, describing the conventional tri-cycle landing gear of the Citation, three submodels are being considered, i.e. those for the vertical forces, the side forces and the longitudinal forces respectively.⁷ These are the forces acting on the aircraft due to ground contact. For the simulation of 'minimum unstuck speed' a tailscraper was added to the model.

In Section 4.5, the final landing gear model, resulting from flight and taxi tests, will be briefly described.

It is possible to implement the a priori airplane models for evaluation with the pilot-in-the-loop. This gives the opportunity to check the general characteristics of these models at an early stage in the model development process.

4. Airplane model identification

4.1 Flight test measurement system

Airplane model identification includes the selection of an airborne measurement system and the specification of the aircraft configuration, flight conditions and maneuvers for system identification.⁸ The measurement system is primarily required to measure the time-histories of input and output variables. In the high accuracy flight test measurement system, which was designed and build by the National Aerospace Laboratory (NLR),⁹ four more-or-less independent sensor systems can be distinguished, see Fig. 8:

- Inertial measurement system

- Air-data measurement system, and α - and β vanes
- Transducers to measure engine parameters
- Transducers to measure control forces and displacements, control surface and trim deflections, landing gear parameters like shock absorber deflections, and nosewheel steering angle

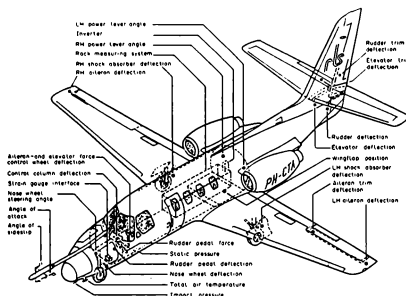


Fig. 8: Citation flight test transducer lay out

A Honeywell laser-gyro Inertial Reference System (IRS) measures specific aerodynamic or ground reaction forces and aircraft body rotation rates and accelerations. High accuracy pressure transducers in a temperature controlled box were used to measure airspeed and altitude variations in quasi steady flight. A variety of other variables was measured in addition, such as primary control forces and displacements, elevator, aileron, rudder and trimtab deflections, angle of attack and side-slip angle, using a specially designed nose boom. Where possible, standard onboard systems were used, such as for the measurement of engine speeds, inter-turbine temperature (ITT) and fuel flow, fuel quantity, stick shaker, anti skid warning, landing gear down-and-up-locks, speed brakes extended/retracted and radar altitude. All these physical quantities were converted into electrical signals, which after signal conditioning are digitized using a Remote Multiplexer Digitizer Unit. Data from modern avionics that complied with the ARINC standard were processed using an ARINC multiplexer. Digitized data were stored on a 14 track magnetic tape recorder with enough capacity to allow continuous recording of all transducer outputs during the complete test flight, from before engine start until after engine shut down. A reference time signal was recorded on a separate track for correlation of the data tracks. Transducer signals were, depending on frequency contents, sampled with sample frequencies ranging from 2 to 50 Hz. However, for vibration measurements, three accelerometers positioned in the cockpit were sampled at a frequency of 256 Hz. Also sound measurements were recorded in another Citation cockpit, using two microphones.¹⁰

The first stage of post flight data processing consisted of adding the time to the measured data, using a frame synchronizer and expressing the recorded data in engineering units. The recorded data were corrected according to the calibration information of the measured channel and properly grouped to be used in the second stage of data analysis concerned with aircraft state reconstruction and mathematical model identification, see Section 4.3.

4.2 Flight test program

The flight test program, carried out with the instrumented Citation, was designed such that flight test maneuvers were evenly distributed in the admissible flight envelope. Two types of flight test maneuvers were applied for model identification:

- quasi-stationary maneuvers for large but slow variations of input and state variables;
- dynamic maneuvers for fast but small variations of input and state variables.

The quasi-stationary maneuvers were performed to evaluate nonlinear effects in the aerodynamic and flight control system models. Examples of quasi-stationary maneuvers were accelerations and decelerations at constant thrust and altitude, 'slow' pull-up-push-down maneuvers, side slips with varying side-slip angles, trimtab excursions at a constant nominal flight condition and wind-up turns. The dynamic maneuvers were carried out to determine the dynamic stability and control characteristics of the aircraft. Examples of dynamic maneuvers were the responses to manually implemented, approximately block-shaped control displacements. These so-called doublets were sequentially implemented on the elevator, ailerons, rudder and left and right power levers, starting from steady, straight flight.

Quasi-stationary and dynamic maneuvers were executed for all relevant and admissible flight conditions and aircraft configuration combinations. In addition, extensive measurements were taken of dynamic engine responses, airplane stalls, aerodynamic ground effects, take off and landing performance and ground handling characteristics.

4.3 System identification techniques

In order to identify the airplane models, described in Section 2, two different types of system identification techniques¹¹ were applied. The first type (type 1) of system identification technique may be characterized as 'model-adjustment technique', see Fig. 9. The idea is to implement a simulation

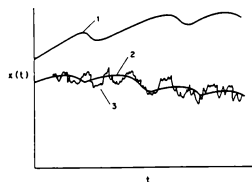


Fig. 9: Principle of model-adjustment technique

model in the computer and to drive the model with the same inputs as the actual system. If the simulated output differs from the output of the actual system, curve 1 in Fig. 9, this must be caused by a difference in initial conditions and/or the parameters in the simulation model, having wrong values. Next, some-usually quadratic-function of the difference between actual and simulated output is minimized with respect to both initial conditions and model parameter values. The technique may be interpreted in terms of statistical estimation theory. The optimal parameter values produce the minimum value of the so-called Likelihood Function and may result in the simulation curve 2 in Fig. 9.

The second type (type 2) of system identification technique is based on explicit mathematical relations, expressing parameter values in terms of system input and output measurements. The so-called 'equation error' methods fall in this category. The explicit mathematical relations follow immediately from regression analysis. In cases where the model structure is not yet precisely known, type 2 identification technique results in a much easier model developments process than type 1 identification technique.

In the present series of flight test measurements, the presence of virtually perfect accelerometers and rate gyro's in the measurement system was fully exploited. The corresponding signals i.e. specific external forces and body rotation rates were interpreted as input signals to a dynamic system model describing the evolution of the aircraft's state.¹² A state reconstruction problem was defined, which was readily solved by applying a type 1 system identification technique. Computed time histories, of for instance airspeed and side slip angle, were matched to corresponding measured time histories by selecting 'optimal' initial conditions and estimating (very small) transducer bias errors. This resulted in an accurate and complete state vector trajectory estimate. As the model was virtually exactly known, the state reconstruction problem needed to be solved only once for each particular flight test maneuver. The main advantage of carrying out a state reconstruction is, that next a type 2 system identification technique may be applied to estimate the aerodynamic derivatives, see Table 1. Herein the simulation models are shown below the system identification technique applied.

System Identification Technique	
Type 1	Type 2
aircraft state trajectory	aerodynamic forces and moments
dynamic engine response	static engine performance
flight control system	flight control system
landing gear side forces	landing gear strut and wheel compression
nose wheel steering	main wheel brakes

Table 1: Classification of identification problems

The performance of this 'two step' method to identify the aerodynamic derivatives is demonstrated in the 'Proof of Match' time histories in Section 5.

4.4 Flight test data analysis

The analysis and modelling of the aerodynamic lift coefficient (C_L) is now described as an example, using the above two-step identification technique. At the start of the analysis it was decided to concentrate in the first instance on the quasi-stationary-horizontal symmetrical reference conditions and the corresponding dynamic maneuvers at the test conditions in the entire flight envelope. The analysis of these test conditions was used both for the evaluation of the performance model as well as for the additional stability and control parts. Fig. 10 presents the test points for the clean configuration in a C_L -Mach plot.

As a reference, also plotted herein are $M^2 C_L$ over corrected aircraft weight curves at four test al-

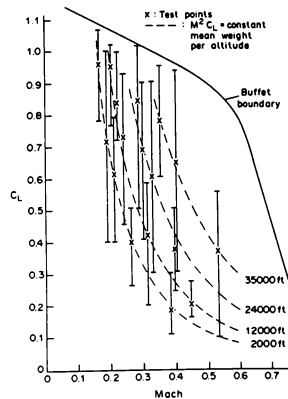


Fig. 10: C_L excursions due to elevator doublets

titudes. The aircraft weight (W) is chosen to represent the mean value of the test weights at that particular altitude. Additionally, in Fig. 10 the C_L excursions as a result of elevator doublets with large amplitude are depicted. The figure also shows the low speed stall and the high speed buffet boundaries for the clean configuration. For each test point in the flight envelope, a submodel was postulated for the three aerodynamic force (C_D , C_L , C_Y) and moment coefficients (C_m , C_q , C_n) in the model axes reference frame, F_{mo} .

The submodel for the lift coefficient, expressed in equation (12), is valid within the normal flight envelope, which means that stall phenomena, buffeting, ground effect, etc. are not represented,

$$C_L = C_{L_0} + C_{L_\alpha} \cdot \alpha + C_{L_\alpha^2} \cdot \alpha^2 + C_{L_q} \cdot \frac{q\bar{c}}{V} + C_{L_{\delta_e}} \cdot \delta_e \quad (12)$$

Because of the size of the aircraft and the speed regime of interest, the aircraft was assumed to be rigid. This eliminated the need for dynamic modelling of additional degrees of freedom due to flexibility.

Fig. 11 shows test points at constant altitude and c.g. position in a C_L - α plot. Because different Mach numbers are attached to each C_L value, Mach effects are embedded. Apart from the reference conditions, the α -sweeps are shown as a result of the elevator doublets. Because the α -sweeps were performed at approximately constant Mach number, Fig. 11 shows that C_L is described by different curves for each aircraft configuration and Mach number.

After integration of the submodels for the lift, drag and pitching moment, it was possible for example to construct C_L/C_D curves for various Mach numbers. Fig. 12 shows these curves for the clean configuration resulting from the analysis. Also

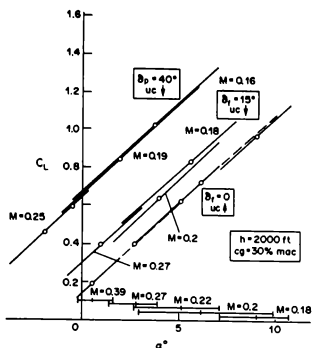


Fig. 11: Different C_L - α curves for various aircraft configurations and Mach

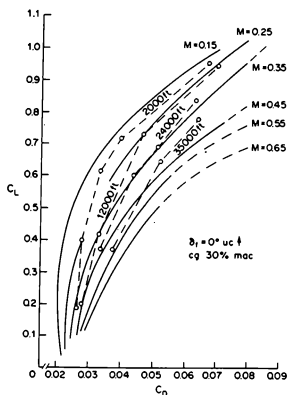


Fig. 12: C_L/C_D curves for various Mach numbers

presented in this figure are the test points at the four flight test altitudes. For all configurations of interest, the coefficients in the submodels were determined using regression analysis and subsequently integrated to $C_L(M)-\alpha$, $C_D(M)-\alpha$ and $C_L(M)-\alpha$ plots for increasing Mach numbers. The clean configuration was considered as a base. Effects due to flap, gear and speedbrake extension were modelled as incremental contributions, which were superimposed on the base model.

As a result of the data processing, a large data set of stability and control derivatives was obtained, valid for the individual test conditions. In accordance with the model for the lift coefficient, these coefficients were formulated as global functions of angle of attack.

4.5 Final Citation 500 models

As a result of the flight test data analysis, the final aerodynamic lift coefficient shows the following modifications, when compared to the a priori lift coefficient, described in Section 3:

In equation (1) the terms $C_{L(\delta_t)}$, see equation (5), and $C_{L(n)}$, see equation (8), could not be identified, so they were deleted. The lift coefficient for the clean configuration as a function of α and M , $C_{L(\alpha, M)}$, equation (3), is shown in Fig. 13, replacing the Figs. 2 and 2A.

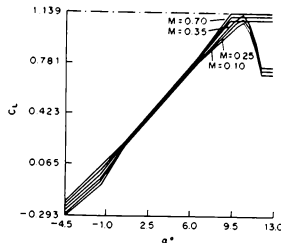


Fig. 13: C_L - α curves for increasing Mach numbers

Fig. 3, presenting $\Delta C_L(\alpha, \delta_f)$, see equation (2), was replaced by Fig. 14.

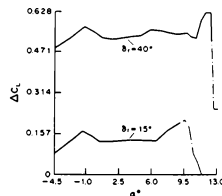


Fig. 14: ΔC_L - α curves for $\delta_f = 15^\circ$, resp. 40°

In equation (4) the terms $\Delta C_{L_{\delta_e}}(M, h)$ and $\Delta C_{L_{\delta_e}}(\delta_e)$ could not be identified, so they were dropped. Fig. 15 shows $C_{L_{\delta_e}}(\alpha, \delta_f)$ in this equation.

In equation (6), $C_{L_q}(M, h)$ was replaced by $C_{L_q}(\alpha, \delta_f)$, see Fig. 16. The term $\Delta C_{L_q}(M, \delta_f)$ was deleted from this equation, as it could not be identified. For operation in the normal flight envelope, the contribution of $C_{L(\dot{\alpha})}$, equation (7), was embedded in the contribution of $C_{L(q)}$, as was already mentioned in Section 3.2. In equation (9) the term $\Delta C_{L_{SB}}(\alpha, \delta_f)$ could not be identified, so it was eliminated. In

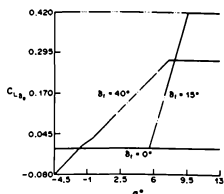


Fig. 15: $C_{L\delta_e}$ - a curves for various flap settings

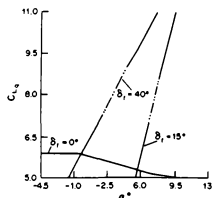


Fig. 16: C_{Lq} - a curves for various flap settings

equation (10) the term ΔC_{LUCM} (a) . M was deleted, but a new term C_{LqUC} (a, δ_f) . $\frac{q}{V}$ was added. Finally, in equation (11) the terms $C_{L\delta_{GE}}$ (δ_f) . $\frac{q}{V}$ and $C_{L\delta_{GE}}$ (a, δ_f) . δ_e could not be identified, so they were left out of this equation.

Results of the identification of the complete aerodynamic model are presented in Section 5 in the form of 'Proof of Match' time histories.

The engine model for the P&W JT15D-1 engines was decomposed into two submodels:

- Static engine thrust model
- Dynamic engine model

For the final static engine performance model, the P1532C computer program was used, see Section 3, complemented with existing testbank data. From a comparison of partial climb recordings of the aircraft and the simulation model, it turned out, that for engine settings above 90% fan rpm (N1) the model showed superior climb performance relative to the aircraft. Because the angle of attack in flight and simulation corresponded very well in these cases, a discrepancy in the engine model was suspected at high power settings. From an analysis of both partial climb and aircraft longitudinal acceleration measurements, a correction was applied to the static thrust performance data at the higher power settings.

The dynamic engine model, which is very important in real-time simulation, was developed starting from a

basic turbofan engine model. Flight test data during throttle chops and slams were available for the analyses of the engine dynamics. From these data the characteristic functions, parameters and time constants in the model were determined. Power change dynamics of the final engine model are demonstrated in the form of a 'Proof of Match' in Section 5.

As the Citation 500 is equipped with a fully manual flight control system (FCS), the aerodynamic hinge moments on the control surfaces must be counter balanced by pilot generated control forces. Therefore a model of the aerodynamic hinge moments was an essential part of the FCS model and the identification¹³ had to be based on measurements in flight. Model development started from a small set of flight test measurements in one flight condition and for one aircraft configuration. Fourth order models of the elevator and rudder FCS, where only the masses of the forward system (control column and rudder pedals respectively) and the aft system (elevator and rudder) were taken into account, were found to result in an adequate fit to the measured control and surface displacements. The aileron FCS required a 6th rather than a 4th order system model, to account for the masses of the control wheel and each of the ailerons. Since the FCS model represents a dynamical system, type 1 identification methods were applied for selection and estimation of the model parameters. In the analysis of the flight test manoeuvres in all other flight conditions and aircraft configurations, the form of the FCS models was kept constant. This analysis resulted in a detailed map of aerodynamic hinge moment derivatives, which was subsequently implemented in the global FCS simulation model of the Citation.

The final landing gear model of the Citation consists of four submodels, i.e. the wheel compression and gear strut model, the landing gear side force model, the main wheel brake model and the nose wheel steering model. Type 1 and type 2 identification techniques were used to identify these landing gear submodels, see Table 1. Due to the availability of the IRS, providing high accuracy time histories of the external specific forces on the aircraft, the body rotation rates, accelerations and attitude angles, type 2 identification techniques could be applied to identify the wheel compression/gear strut and main wheel brake models. The landing gear side force model⁷ could be identified using type 1 identification techniques on a dynamical model, wherein the wheel slip angles were explicitly calculated. As mentioned already, the nose wheel steering system of the Citation is connected via springs to the rudder pedals. Type 1 identification techniques were applied to identify second and fourth order dynamical nose wheel steering models. However, it turned out, that a quasi-static model performed surprisingly well. In the final landing gear simulation model, the tyre and strut deflections and rates of deflection are computed first, resulting in the vertical forces. These forces play a crucial role in the computation of the side forces and the longitudinal landing gear forces. Parameters related to the condition of the runway surface may have large effects on the magnitudes of these forces and consequently on the landing gear behaviour. As the ground measurements during taxi tests were performed on dry concrete, no effects of runway condition could be determined. Therefore, relevant data from available literature⁷ was incorporated in the model. To prevent unrealistic behaviour of the simulated land-

ing gear at zero aircraft speed, so called zero-speed models were added to the brake force, the side force and the nose wheel steering models.

5. Comparison of a priori and final model responses

5.1 General

A quantitative validation of the quasi-stationary and dynamic characteristics of the a priori as well as the final Citation models is possible, using the 'Proof of Match' (POM). In the POM simulated airplane responses are directly compared to aircraft flight traces providing for an objective examination of the mathematical model fidelity. For the generation of POM-data the data-files resulting from flight tests are read from the flight test tapes. In this manner the control forces and displacements, control surface position and power lever angles, as measured by the flight test measurement system, are used as the input signals to the six Degrees of Freedom (DOF) engineering simulation⁴ of the Citation. The responses on these input signals are plotted against the same responses as measured in the aircraft during special flight test maneuvers.¹⁴ The next Section describes the POMs of the a priori and the final Citation 500 models.

5.2 A priori versus final Citation 500 model responses

In order to compare the a priori and final mathematical models and data of the Citation 500, POMs of both models were generated and subsequently plotted in one figure. By taking modelling data from the final into the a priori models, the cause of the differences between the responses of both models was examined. For the a priori models corrected in this manner, new POMs were generated in order to check if the correction made sense. Examples of the POMs of the a priori, the corrected a priori, and final models are presented below.

5.3 POM comparisons

POM comparisons are presented for the following models:

- the aerodynamic model
- the engine model
- the integrated Citation 500 simulation model

The POM comparisons of the aerodynamic models concern the airplane 'short period dynamics' with full flaps and gear down, and the 'roll response' in the clean configuration. Fig. 17 shows the pitch response (THETA) due to an elevator step-input and release after two seconds.

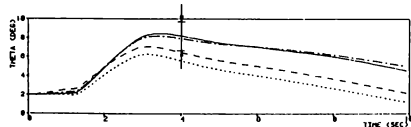


Fig. 17: Short period motion (full flaps, gear down)

— measured aircraft response
 a priori model
 ----- 'corrected a priori'
 -.-.-.- 'final' model

Comparing the pitch response of the a priori model (dotted line) with the measured aircraft response shows a discrepancy which is out of FAA Phase II tolerance. Taking the change in lift coefficient due to full landing gear extension $C_{L_{UC}}$ (a, δ_P), eq. (10),

from the final aerodynamic model, results in an improved response of the 'corrected a priori model' (dashed line). The final aerodynamic model response fits the measured response almost perfectly.

Fig. 18 shows the roll rate response (P) due to an aileron block-shaped input during five seconds.



Fig. 18: Roll rate response (clean)

Comparing the roll rate response of the a priori model (dotted line) with the measured aircraft response shows a discrepancy which is just out of FAA Phase II tolerance. Taking the increment in rolling moment coefficient due to roll rate ($C_{l(p)}$)

from the final aerodynamic model, results in a roll rate response of the 'corrected a priori model' (dashed line) which is within FAA Phase II tolerance. Also here the final aerodynamic model response fits the measured response almost perfectly.

The POM comparison of the engine model concerns the 'power change dynamics' of the P & W JT15D turbofan engine. Fig. 19 shows the change in Net Thrust (TN1) due to a power lever angle (PLA) change from 32° to idle in two seconds.

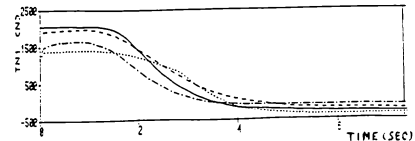


Fig. 19: Power change dynamics (Net Thrust)

Comparing the response of the a priori engine model (dotted line) with the measured engine response shows a significant difference in the starting value of the thrust at 32° PLA. The a priori engine model dynamics are not correct either, however the thrust at idle power is correct. In the 'corrected a priori model' (dashed line) the table of HP rotor speed has been adapted to the measured stationary starting value. Moreover the engine model time-constant (τ) has been halved. The figure shows that the 'corrected a priori model' response is now closer to the measured engine response than the 'final model' response (dot-dash line).

The POM comparison of the integrated Citation 500 model concerns the 'normal take-off'. Fig. 20 shows the aircraft pitch angle (θ) and radio altitude (H-RADIO) during this maneuver.

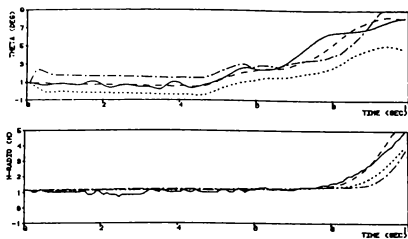


Fig. 20: Normal take-off

Comparing the pitch response of the a priori model (dotted line) with the measured aircraft pitch response shows a significantly smaller pitch angle of the a priori model. The response of radio altitude shows that the a priori Citation leaves the runway too late. Taking the values for the length of the nose gear, the change in lift coefficient due to ground effect, $C_{L(GE)}$ (eq. 11), and the change in pitching moment coefficient $C_{m(GE)}$ due to ground effect from the final Citation model, results in an almost perfect response of the 'corrected a priori model' (dashed line). This response is even better than the final model response which sticks to the runway even longer than the a priori model. Reason is that not all measured take off data were used (yet) for model identification, and so were not used (yet) for the final model.

6. Concluding remarks

DATCOM techniques were compared to flight test results in composing a flight simulation model. On the basis of this comparison, it has been shown that quite acceptable a priori models can indeed be composed using inexpensive DATCOM techniques. Many model improvements were possible, however, by using the results of an extensive flighttest program. These model improvements were thought to warrant the extra costs of the flighttest program. Further more, since FAA Phase II/III simulator approval requires a successful Proof of Match, flight testing is necessary anyway. The balanced use of both techniques in the composition of flight simulation models may lead to smaller flighttest programs and reduced costs of flight validated models. 'Low cost' flight simulators for commuter airlines and general aviation training may benefit from this development.

7. References

- Williams, J.E., Vulkelich, S.R. (1979). The USAF Stability and Control Digital Datcom, Vol. I, Users Manual, Technical Report AFFDL-TR-79-3032, Vol. I. McDonnell Douglas Astronautic Company, St. Louis Missouri 63166, USA.
- McDonnell Douglas Corp. (1976). USAF Stability and Control Datcom. Air Force Flight Dyn. Lab., U.S. Air Force.
- Smetana, F.O. (1984). Computer Assisted Analysis of Aircraft Performance Stability and Control. McGraw-Hill Book Company, ISBN 0-07-58441-9.
- Baarspul, M. (1990). A review of flight simulation techniques. In: Progress in Aerospace Sciences, Vol. 27:1, Pergamon Press, Oxford, UK.
- IATA (1986). Flight Simulator Design and Performance Data Requirements (Final draft of proposal revision A).
- Baarspul, M., Dooren, J.P. van, (1976). The Hybrid Simulation of Aircraft Motions in a Piloted Moving-base Flight Simulator. Report VTH- 178, Delft University of Technology, Delft, The Netherlands.
- Wahi, M.K., Yourkowski, F.M. (1980). 757-200 crew training simulator-ground handling data. Boeing Commercial Airplane Company, Seattle, USA.
- Klein, V. (1989). Estimation of aircraft aerodynamic parameters from flight data. In: Progress in Aerospace Sciences, Vol. 26:1, Pergamon Press, Oxford, UK.
- Nieuwpoort, A.M.H., Breeman, J.H., Baarspul, M., Mulder, J.A. (1988). Phase II Flight Simulator Mathematical Model and Data-Package, Based on Flight Test and Simulation Techniques. In: ICAS Proceedings 1988, Vol. 2, Paper ICAS-88-1.9.2, 16th Congress of the International Council of the Aeronautical Sciences, Jerusalem, Israel.
- Ruijgrok, G.J.J., Paassen, D.M. van, (1987). Sound Measurements of the Cessna Citation PH-CTE. Memorandum M-565, Delft University of Technology, Delft, The Netherlands.
- Eijkhoff, P. (1974). System identification. John Wiley & Sons, ISBN 0-24980-7.
- Mulder, J.A. (1987). Design and Evaluation of Dynamic Flight Test Manoeuvres. Report LR-497, Delft University of Technology, Delft, The Netherlands.
- Mulder, J.A. (1988). Aircraft flight control system identification. In: Proceedings of 8th IFAC-IFORS Symposium on Identification and System Parameter Estimation, Beijing, China.
- Anon. (1987). Citation 500 aircraft pilot test procedures for FAA Phase II ATG manoeuvres, Revision 1, CAE Electronics Ltd., Montreal, Canada.

HOT-BENCH SIMULATION OF THE ACTIVE FLEXIBLE WING WIND-TUNNEL MODEL

AIAA-90-3121-CP

Carey S. Buttrill* and Jacob A. Houck**
NASA Langley Research Center
Hampton, Virginia 23665-5225

Abstract

Two simulations, one batch and one real-time, of an aeroclastically-scaled wind-tunnel model were developed. The wind-tunnel model was a full-span, free-to-roll model of an advanced fighter concept. The batch simulation was used to generate and verify the real-time simulation and to test candidate control laws prior to implementation. The real-time simulation supported hot-bench testing of a digital controller, which was developed to actively control the elastic deformation of the wind tunnel model. Time scaling was required for hot-bench testing. The wind-tunnel model, the math models for the simulations, the techniques employed to reduce the hot-bench time-scale factor, and the verification procedures are described.

Nomenclature

a_g	first-order pole of typical actuator transfer function, rad/sec
$[A_k^{ff}]$	$n_f \times n_f$ matrix where $A_{00}^{ff}(i,j) = \frac{\partial(-F^{n_i}/\bar{q})}{\partial \eta_j^{(k)}}$, negative force coefficient on the i^{th} elastic mode due to k^{th} time derivative of j^{th} elastic mode, $k = 0, 1, 2$
$[A_{m+2}^{ff}]$	$n_f \times n_f$ matrix where $A_{m+2}^{ff}(i,j) = -\frac{\partial(-\dot{x}_{im}^f)}{\partial \eta_j^{(k)}}$, effect of j^{th} elastic mode rate on the derivative of the m^{th} unsteady aerodynamic lag state associated with the i^{th} flexible mode, $m = 1, \dots, n_{lag}$
$[A_k^{fc}]$	$A_k^{fc}(i,j) = \frac{\partial(-F^{n_i}/\bar{q})}{\partial \delta_j^{(k)}}$, force on i^{th} elastic mode due to k^{th} time derivative of j^{th} control mode, $k = 0, 1, 2$
$[A_k^{cf}]$	$A_k^{cf}(i,j) = \frac{\partial(-F^{\delta_i}/\bar{q})}{\partial \eta_j^{(k)}}$, opposing hinge moment on i^{th} control mode due to k^{th} time derivative of j^{th} elastic mode
$[A], [B]$ $[C], [D]$	general continuous system dynamic matrices
\bar{c}	wing chord, 39.76 in
F^{n_i}	total generalized force on elastic (flexible) mode i
$[F], [G]$	general discrete system dynamic matrices

* Senior Member, AIAA

** Associate Fellow, AIAA

$[G_0]$	discrete system input matrix applied to $\{u_k\}$
$[G_1]$	discrete system input matrix applied to $\{u_{k+1}\}$
$[G^f]$	diagonal modal damping matrix, $G^f(i,i) = .03$
h	integration step size, seconds
HM_δ	control surface hinge moment, in-lbs
k_δ	actuator steady state gain
$[K^f]$	diagonal modal stiffness matrix, $K^f(i,i) = m_i \omega_i^2$
$[M^f]$	diagonal modal mass matrix, $M^f(i,i) = m_i$
$[M^{cf}]$	matrix of control mode to elastic mode inertial coupling
\bar{q}	dynamic pressure, lbs/in ²
n	rate limit
s	Laplace variable
$\{u_k\}$	vector input at time $t=kh$
V	airspeed, in/sec (6696 in/sec at Mach = .5)
x	state
$\{x_k\}$	state vector at time $t=kh$
δ_i	generalized coordinate - control mode i
ζ_δ	damping of second-order denominator term of typical actuator transfer function, rad/sec
η_i	generalized coordinate - elastic mode i
ξ_g	output of Dryden turbulence transfer function
ρ	density, 1.146×10^{-7} lbs-sec ² /in ⁴ or slinches/in ³
σ_g	RMS turbulence intensity, in/sec
v	Gaussian random number
ω_δ	frequency of second-order denominator term of typical actuator transfer function, rad/sec
ω_g	break frequency of turbulence transfer function, $(2\pi)(17.32)$ rad/sec
ω_{aa}	break frequency of anti-aliasing transfer function, $(2\pi)(25.0)$ rad/sec
$[]$	matrix
$\{\}$	column vector

Subscripts

a	aerodynamic
as	anti-symmetric
c	commanded
lag	aerodynamic "lag" quantity
lin	linear
neg	negative
pos	positive
nl	no load
STALL	actuator dynamic stall value
sy	symmetric
0	quantity associated with position
1	quantity associated with rate
2	quantity associated with acceleration

Superscripts

f	associated with elastic (flex) mode equations
ff	effect on elastic due to elastic (flex-flex)
fc	effect on elastic due to control (flex-control)
fg	effect on elastic due to gust input (flex-gust)
c	associated with control mode equations
cf	effect on hinge moment due to flex (control-flex)
cc	effect on hinge moment due to control
cg	effect on hinge moment due to gust input

Abbreviations

AB2	Adams-Borthforth second-order predictor
AFW	Active Flexible Wing
CAMAC	Computer Automated Measurement and Control
ISAC	Interaction of Structures, Aerodynamics and Controls
LaRC	Langley Research Center
LEI	Leading Edge Inboard
LEO	Leading Edge Outboard
PPU	Peripheral Processor Unit
psf	pounds per square foot
RK2	Runge-Kutta second-order predictor-corrector
RMS	root-mean-square
RVDT	Rotary Variable Differential Transducer
TDT	LaRC Transonic Dynamics Tunnel
TEI	Trailing Edge Inboard
TEO	Trailing Edge Outboard

Introduction

The simulations described in this paper were developed as part of the ongoing Active Flexible Wing (AFW) Wind-Tunnel Test Program,^[1,2,3] a collaborative effort between NASA Langley Research Center (LaRC) and Rockwell International Corporation. Three wind-tunnel tests have been completed and another is scheduled for February of 1991. The program objective is the validation of analysis and synthesis methodologies as applied to the multi-function control of a sophisticated aeroelastic wind-tunnel model. The control functions being investigated include suppression of flutter, roll performance maximization, and load alleviation. Only the simulation models developed to support flutter suppression are discussed in this paper. Flutter is a potentially explosive dynamic instability that can occur when a sufficiently flexible wing begins to extract energy from the fluid stream. During the most recent tunnel entry, completed in November 1989, flutter suppression was successfully demonstrated at a dynamic pressure 24 percent above a measured open-loop flutter point.^[2,3]

Simulation Roles

Two distinct, but interrelated, simulations were developed to support the AFW test program, a batch simulation and a real-time simulation. The batch simulation served as a "truth" model, and was used to: (1) evaluate the control laws to predict performance and establish gain and phase margins; (2) provide models and data files for the real-time hot-bench simulation; and (3) verify the real-time simulation.

The real-time simulation of the model/wind-tunnel environment served to verify the functionality of the digital controller system in a hot-bench laboratory. End-to-end verification and debugging of the complex, one-of-a-kind

digital controller system was essential, since whenever flutter testing is undertaken, both the model and the tunnel are at risk.

Wind-Tunnel Model

Figure 1 shows the AFW model mounted in the LaRC Transonic Dynamics Tunnel (TDT) during the November 1989 test. The sting mount has an internal ball bearing arrangement that allows the model to roll $\pm 145^\circ$ about the sting axis. The fuselage is connected to the sting with a hydraulically powered pivot so that the model can be remotely pitched from approximately -1.5° to $+13.5^\circ$. Destabilizing mass ballast was added to each wingtip so that the model would flutter within the operating envelope of the TDT^[2]. The tip ballast serves to lower both the first-bending and the first-torsion elastic mode frequencies, with the predominant effect on the first-torsion mode. The result is that the first-torsion and the first-bending elastic modes combine to form the primary flutter mechanism at a lower dynamic pressure than was the case for the original model (with no tip ballast). The tip ballast can also be rapidly decoupled in pitch from the wingtip by releasing a hydraulic brake. When decoupled, the tip ballast is restrained in pitch by a soft spring. Decoupling the tip ballast proved to be an effective flutter stopper during testing, providing a significant safety margin. For the flutter suppression tests conducted in the November 1989 test, the

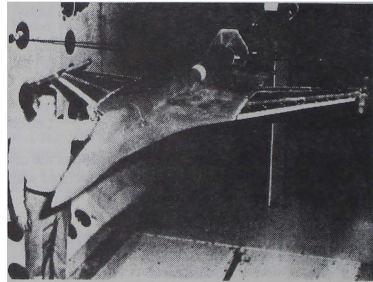


Figure 1. - AFW model mounted in the LaRC Transonic Dynamics Tunnel.

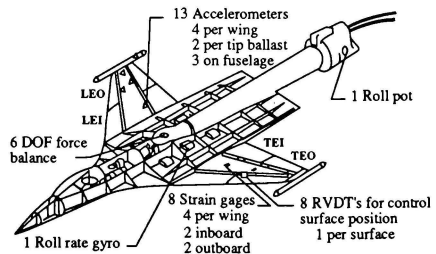


Figure 2. - Instrumentation of the AFW model.

roll brake was engaged, so that the model was not free to roll.

Figure 2, a drawing of the model, illustrates the locations of the control surfaces and selected instrumentation. There are eight control surfaces and thirteen accelerometers. With the addition of the tip ballast, two wing accelerometers, previously co-located with the leading edge inboard control surfaces, were moved to the tip ballasts. Reference 4 describes in detail the AFW wind-tunnel model prior to adding the tip ballast.

Hot-Bench Laboratory

The AFW hot-bench simulation set-up, depicted schematically in figure 3, utilized the central real-time facility at LaRC^[5]. The LaRC real-time facility consists of nodes or simulation sites that communicate by means of a 50-megabit-per-second fiber-optic digital-data network. The various simulation nodes on the network included two Control Data Cyber 175 computers, engineering control consoles, various aircraft cockpits, and motion base hardware. For the AFW hot-bench simulation, one Cyber 175 was used to integrate the equations of motion. An Adage graphics computer, used in this study, communicates directly with a Cyber 175 through a port. New Terabit Eagle 1000 graphics computers, currently being installed at LaRC, will communicate over the fiber-optic network as another simulation node. Communications with the digital controller occurred over analog lines in the same manner as when the controller was connected to the wind-tunnel model at the LaRC TDT.

Real-Time Graphics

To assist in visualizing the simulated model dynamic behavior, a real-time display (figure 4) was developed on an ADAGE graphics computer. The display presents model

pitch and roll, control surface deflections, and total structural deformation of the simulated wind-tunnel model. The display is based on a finite-element structural model of the test article. A subset of the finite-element structural nodes were used to highlight the main body, wings, and the eight control surfaces. The dashed line represents the undeformed configuration. The deformations can be exaggerated for ease of viewing by a factor set at the console.

Simulation Time Scaling

The AFW hot-bench simulation operated at a time scale slower than 1:1 (real time). Time-scale is a function of the integration step (h) and the computer frame (T). If T is larger than h , the simulation runs at $1:(T/h)$ "slow." Since there was no human operator in the hot-bench loop, a time-scale other than 1:1 could be accommodated. Several factors prevented the AFW hot-bench simulation from operating in real time. The control computer was scheduled to run at 200 Hz in the wind-tunnel. To avoid excessive digitally-induced time delays, the hot-bench simulation needed to update at least twice for each digital controller frame, requiring a 400 Hz rate for the simulation if it were to run in real time. The minimum frame time available on the Cyber 175 real-time clock was 5 milliseconds (200 Hz). The simulation model itself was sufficiently complex that a computational frame of at least 12.5 milliseconds (80 Hz) was required. Furthermore, since there are only two Cyber real-time computers and many real-time jobs, the hot-bench simulation often shared a Cyber 175 with another job. With only one half of the Cyber computing power available, the 80 Hz simulation update rate would be further reduced to 40 Hz. Time-scale must, therefore, be an easily adjusted parameter for any dynamic component in the hot-bench loop.

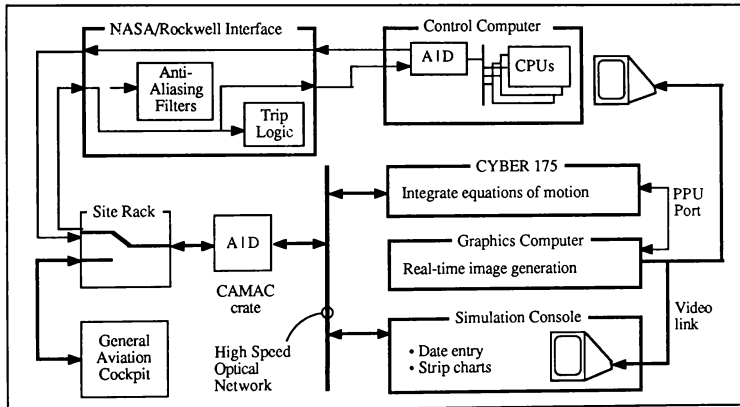


Figure 3. - Schematic of the AFW hot-bench simulation laboratory.

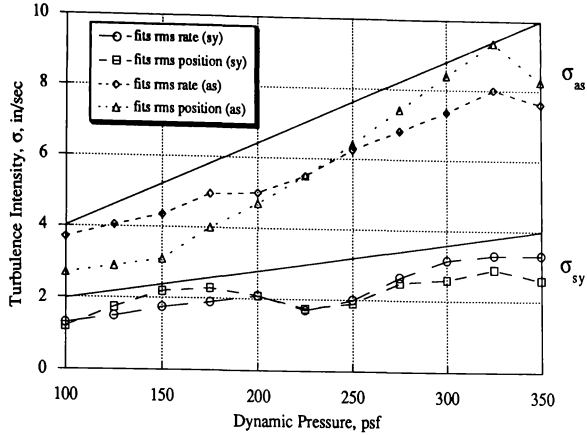


Figure 5. - Turbulence intensities used for simulation.

follows,

$$\dot{x}_{0lin} = k_{\delta} a_p (\delta_c - x_0)$$

Positive and negative rate limits based on no-load rate limits modified by hinge moment were formed,

$$r_{lpos} = (r_{nl}) \sqrt{1 - (HM_{\delta} / HM_{\delta_{STALL}})}$$

$$r_{lneg} = - (r_{nl}) \sqrt{1 + (HM_{\delta} / HM_{\delta_{STALL}})}$$

where the hinge moment, HM_{δ} , is positive for an external load resisting positive actuator motion, and $HM_{\delta_{STALL}}$ represents the maximum load the actuators can overcome. Note that an aiding load will produce a rate limit larger than the no-load rate limit, r_{nl} . An aiding load could occur for a leading edge surface. The positive and negative limits were imposed on the linear rate \dot{x}_{0lin} as follows,

$$\dot{x}_0 = \begin{cases} r_{lpos} & \text{if } \dot{x}_{0lin} > r_{lpos} \\ \dot{x}_{0lin} & \text{otherwise} \\ r_{lneg} & \text{if } \dot{x}_{0lin} < r_{lneg} \end{cases}$$

The state x_0 was then used as a command to the second-order portion of the actuator transfer function

$$\ddot{\delta} = \omega_g^2 (x_0 - \delta) - 2 \zeta_g \omega_g \dot{\delta}$$

Once the position, velocity and acceleration of the control surfaces were obtained, they were resolved into symmetric

and anti-symmetric components that became inputs into the aeroelastic equations.

Turbulence Model

A turbulence model based on the Dryden atmospheric turbulence model^[9] was used. A break frequency of 17.23 Hz for the turbulence transfer function was used to approximate the expected wind-tunnel turbulence. Resonance peaks at 10 Hz were observed in tunnel data from prior entries and 10-11 Hz was the predicted flutter frequency. A break frequency of 17.23 Hz produces a peak magnitude at 10 Hz in the Dryden transfer function. The state equations used for each symmetry are,

$$\ddot{x}_g = -2 \omega_g \dot{x}_g - \omega_g^2 x_g + \sigma_g \omega_g^{3/2} (T_v^{-1/2} v)$$

where, v is a Gaussian random process that is sampled and held every T_v seconds, ω_g is the break frequency in radians/second, and σ_g is the root-mean-square (RMS) intensity. In both the batch and hot-bench simulations, T_v is the integration step size. The factor of $T_v^{-1/2}$ that multiplies the random input arises from the fact that a digital process is inherently band-limited. The random processes (one for each symmetry) that produces v can be interpreted as white noise being passed through a perfect $\frac{1}{2T_v}$ Hz band-pass filter. The output equations are,

$$\xi_g = x_g + \frac{\sqrt{3}}{\omega_g} \dot{x}_g \quad \text{and} \quad \dot{\xi}_g = \dot{x}_g + \frac{\sqrt{3}}{\omega_g} \ddot{x}_g$$

Prior to the November 1989 entry, an overall intensity level of 12 in/sec was estimated for the tunnel turbulence. It was decided to allocate the turbulence between the symmetric and anti-symmetric models based on an 85/15 percent distribution. The following symmetric and anti-symmetric intensities resulted

$$\sigma_{gsy} = 10.2 \text{ in/sec} \quad \sigma_{gas} = 1.8 \text{ in/sec}$$

When the model is in the tunnel with a flutter-suppression control law engaged and subjected only to natural turbulence as excitation, control surface activity results. Analysis of data generated in the November 1989 tunnel test revealed that the RMS control surface activity predicted by simulation was much higher than actually experienced in the tunnel. Three different flutter control laws were tested in the November 1989 tunnel test and all generally resulted in both the same observed RMS levels and the same degree of overprediction by the simulation. By making the turbulence intensities (σ_g) functions of dynamic pressure, simulation-predicted RMS levels can be brought into agreement with observed data. An example of this process is shown in figure 5 for the flutter suppression control law that achieved the highest dynamic pressure. The intensities required to bring batch-simulation-generated RMS results for both commanded control positions and control rates into agreement with experimental data are plotted in figure 5 as functions of dynamic pressure. For each symmetry, one intensity function corrects the RMS rate predictions and the other corrects the RMS deflection predictions. Since only one intensity exists per symmetry, the RMS rate results and the RMS position results cannot both be matched. The solid lines (one per symmetry) on figure 5 represent a fitted line biased upwards. The upward bias favors overprediction, which is conservative and preferable, if small. The solid lines on figure 5 are given by the following functions,

$$\sigma_{gsy} = 0.4 + 1.0 \left(\frac{\bar{q}}{100} \right) \quad \sigma_{gas} = 1.6 + 2.4 \left(\frac{\bar{q}}{100} \right)$$

where \bar{q} has units of psf. These estimates should be regarded as an incremental improvement, rather than a final characterization of turbulence in the TDT.

Aeroelastic Equations

The aeroelastic equations in a frequency domain format are given by [8,10,11]

$$\begin{aligned} & \left(-\omega^2 \begin{bmatrix} [M^f] & [M^c] \\ [M^c] & [M^e] \end{bmatrix} + j\omega \begin{bmatrix} [C^f] & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} [K^f] & 0 \\ 0 & 0 \end{bmatrix} \right) \begin{Bmatrix} \eta \\ \delta \end{Bmatrix} \\ & + \rho \left(\frac{V^2}{2} \right) \begin{bmatrix} [Q^{ff}(j\omega)] & [Q^{fc}(j\omega)] \\ [Q^{cf}(j\omega)] & [Q^{cc}(j\omega)] \end{bmatrix} \begin{Bmatrix} \eta \\ \delta \end{Bmatrix} \\ & = -\rho \left(\frac{V^2}{2} \right) \begin{Bmatrix} Q^f(j\omega) \\ Q^c(j\omega) \end{Bmatrix} \begin{Bmatrix} \xi_B \\ \xi_V \end{Bmatrix} \end{aligned} \quad (1)$$

These equations consist of the standard in-vacuo second-order matrix structural equations (mass, damping, and stiffness)

and apply to either symmetric or anti-symmetric motion. The flexible modes are augmented with control modes that represent control surface deflection. The control modes have zero stiffness. A low frequency subset of the elastic modes of free vibration from a large-order structural model are typically used in an aeroelastic formulation. For the AFW simulations, the number of retained flexible modes per symmetry was between seven and ten. The flexible modes are orthogonal to each other so the mass and stiffness matrices are diagonal. Modal damping of 0.03 is assumed for each mode.

The in-vacuo equations were augmented with generalized aerodynamic force coefficient matrices, $[Q(j\omega)]$, that are functions of frequency. The functions, $[Q(j\omega)]$, can be approximated^[10,11] by matrix expressions that are rational in the Laplace variable, s . The "least square"^[10,11] form of approximation is given by

$$\begin{aligned} [\hat{Q}(s)] &= [A_0] + [A_1]ts + [A_2](ts)^2 \\ &+ \sum_{m=1}^{n_{lag}} [A_{2+m}] \left(\frac{ts}{ts + \beta_m} \right) \end{aligned} \quad (2)$$

where $\tau = (c/2V)$ and n_{lag} is between one and four, depending on the order of the fit. Equation (1) and (2) can be combined to produce the time domain aeroelastic equations (3) in table 1, wherein the second-order in-vacuo equations were augmented with unsteady aerodynamic "lag" states arising from the denominator term being summed in equation (2). Control surface positions, rates, and accelerations along with turbulence are treated as external inputs. The vectors $\{\eta\}$ and $\{\dot{x}_{am}\}$ are both $n_f \times 1$, where n_f is the number of retained elastic modes. Equations (3) are used to solve for the elastic mode accelerations, $\{\ddot{\eta}\}$, which can be integrated to find the rates and displacements. Equations (4) in table 1 are used to calculate actuator hinge moments. A positive hinge moment in this case resists positive actuator motion. The derivative calculations indicated in equations (3) and (4) were performed for each symmetry in the simulations. The symmetric and anti-symmetric components of the final accelerometer outputs were resolved into right and left components before output.

Anti-Aliasing Filters

For all the simulations, the dynamics of the anti-aliasing filters on the 40 primary outputs were simulated. For the tunnel test, both single-pole filters with a break frequency of 25 Hz and fourth-order Butterworth filters with break frequencies of 100 Hz, had been assembled and were available. Only the single-pole filters were used in the November 1989 tunnel test. The single pole anti-aliasing filters are given by,

$$\frac{x(s)}{u(s)} = \frac{1}{(s/\omega_{aa}) + 1}$$

where ω_{aa} is the break frequency in radians/second.

Table 1 Aeroelastic and Hinge Moment Equations

$$\begin{aligned}
 & \left([M^f] + \rho \left(\frac{\bar{c}^2}{8} \right) [A_2^f] \right) \left\{ \begin{matrix} \bar{\eta} \\ \bar{\delta} \end{matrix} \right\} + \left([G^f] + \rho \left(\frac{\bar{c}V}{4} \right) [A_1^f] \right) \left\{ \begin{matrix} \dot{\eta} \\ \dot{\delta} \end{matrix} \right\} + \left([K^f] + \rho \left(\frac{V^2}{2} \right) [A_0^f] \right) \left\{ \begin{matrix} \eta \\ \delta \end{matrix} \right\} - \rho \left(\frac{\bar{c}V}{4} \right) \left(\left\{ \begin{matrix} x_{a1}^f \\ \vdots \\ x_{a4}^f \end{matrix} \right\} \right. \\
 & \quad \left. - \rho \left(\frac{\bar{c}V}{4} \right) \left\{ A_1^f \right\} \left(\frac{\xi_g}{V} \right) - \rho \left(\frac{V^2}{2} \right) \left\{ A_0^f \right\} \left(\frac{\xi_g}{V} \right) \right. \\
 & \quad \left. - \left([M^{fc}] + \rho \left(\frac{\bar{c}^2}{8} \right) [A_2^{fc}] \right) \left\{ \begin{matrix} \bar{\delta} \\ \bar{\delta} \end{matrix} \right\} - \left(\rho \left(\frac{\bar{c}V}{4} \right) [A_1^{fc}] \right) \left\{ \begin{matrix} \dot{\delta} \\ \dot{\delta} \end{matrix} \right\} - \left(\rho \left(\frac{V^2}{2} \right) [A_0^{fc}] \right) \left\{ \begin{matrix} \delta \\ \delta \end{matrix} \right\} \right. \\
 & \quad \left. \left\{ \begin{matrix} x_{am}^f \\ x_{am}^f \end{matrix} \right\} + \beta_m \left(\frac{2V}{\bar{c}} \right) \left\{ \begin{matrix} x_{am}^f \\ x_{am}^f \end{matrix} \right\} + \left([A_{m+2}^f] \quad [A_{m+2}^{fc}] \right) \left\{ \begin{matrix} \dot{\eta} \\ \dot{\delta} \end{matrix} \right\} = - \left\{ A_{m+2}^f \right\} \left(\frac{\xi_g}{V} \right) \quad (m=1,4)
 \end{aligned} \tag{3}$$

$$\begin{aligned}
 \{HM_6\} &= \left[[M^{cf}] + \rho \left(\frac{\bar{c}^2}{8} \right) [A_2^{cf}] \quad \rho \left(\frac{\bar{c}^2}{8} \right) [A_2^{cc}] \right] \left\{ \begin{matrix} \bar{\eta} \\ \bar{\delta} \end{matrix} \right\} + \left[\rho \left(\frac{\bar{c}V}{4} \right) [A_1^{cf}] \quad \rho \left(\frac{\bar{c}V}{4} \right) [A_1^{cc}] \right] \left\{ \begin{matrix} \dot{\eta} \\ \dot{\delta} \end{matrix} \right\} \\
 &+ \left[\rho \left(\frac{V^2}{2} \right) [A_0^{cf}] \quad \rho \left(\frac{V^2}{2} \right) [A_0^{cc}] \right] \left\{ \begin{matrix} \eta \\ \delta \end{matrix} \right\} - \rho \left(\frac{\bar{c}V}{4} \right) \left(\left\{ \begin{matrix} x_{a1}^c \\ \vdots \\ x_{a4}^c \end{matrix} \right\} + \left\{ \begin{matrix} x_{a1}^c \\ \vdots \\ x_{a4}^c \end{matrix} \right\} + \rho \left(\frac{\bar{c}V}{4} \right) [A_1^{cg}] \left(\frac{\xi_g}{V} \right) + \rho \left(\frac{V^2}{2} \right) [A_0^{cg}] \left(\frac{\xi_g}{V} \right) \right. \\
 & \quad \left. \left\{ \begin{matrix} x_{am}^c \\ x_{am}^c \end{matrix} \right\} + \beta_m \left(\frac{2V}{\bar{c}} \right) \left\{ \begin{matrix} x_{am}^c \\ x_{am}^c \end{matrix} \right\} + \left([A_{m+2}^{cf}] \quad [A_{m+2}^{cc}] \right) \left\{ \begin{matrix} \dot{\eta} \\ \dot{\delta} \end{matrix} \right\} = - \left\{ A_{m+2}^{cg} \right\} \left(\frac{\xi_g}{V} \right) \quad (m=1,4)
 \end{aligned} \tag{4}$$

Pre-Test and Post-Test Models

When the initial batch simulation was being developed, it was assumed the test would be conducted in Freon in the .8 to .9 Mach range, and a real possibility existed of holding Mach constant during a test run and bleeding in Freon to raise the density of the test medium. It was also expected that some of the control laws would be scheduled on dynamic pressure in which case both the batch and hot-bench simulations would need to be able to vary dynamic pressure during a run to

effectively test the control laws. The aerodynamic data arrays in the simulation are strong functions of Mach in the .8 to .9 Mach range. To interpolate each element of the aerodynamic data arrays according to Mach would require excessive CPU time. Therefore, the approach used in the both the pre-test batch and hot-bench simulations was to leave Mach and airspeed fixed and to vary density to achieve a change in dynamic pressure.

Table 2 Pre-Test and Post-Test Simulation Math Models

State Categories	Pre-test (1-lag)	Post-test (4-lag)
Symmetric flexible mode positions and velocities	16	20
Sym aero lag states associated w/ the flexible modes	8	40
Sym aero lag states associated w/ the control modes	4	16
Symmetric turbulence states	4	2
Anti-sym flexible mode positions and velocities	14	20
Anti-sym aero lag states associated w/ the flexible modes	7	40
Anti-sym aero lag states associated w/ the control modes	4	16
Anti-sym turbulence states	2	2
Total aeroelastic states	57	156
Actuator states, 3 per actuator, 8 actuators	24	24
Anti-aliasing filters on 40 channels	40	40
Total states	121	220

Some months prior to the 1989 tunnel entry, the use of Freon as a test medium was forbidden, and it was determined that the test would be conducted with air as the test medium in the .2 to .5 Mach range. In this Mach range the aerodynamic matrices are virtually constant with respect to Mach. Furthermore, in the actual wind-tunnel test, the tunnel was not evacuated to any degree. Dynamic pressure, Mach, and airspeed were all changing as the fan speed was gradually increased. In the post-test implementation, density was left fixed, while airspeed was varied to replicate a given TDT dynamic pressure.

As seen in table 2, the number of states increases in going from the pre-test to the post-test simulation math models. The post-test model uses 20 elastic modes instead of 15. For the post-test model, a 4-lag least-square aerodynamic fit is employed ($n_{lag}=4$) instead of the 1-lag fit used in the pre-test model. As seen in equations (3) and (4) and table 2, the choice of n_{lag} has a dramatic impact on the number of states.

Batch Simulation Implementation

The structure of the batch simulation is identical to the simulation math model as described by differential equations. The state derivatives are all calculated explicitly from the states (outputs of integrators). The state derivatives are collected in a vector and integrated with a Runge-Kutta second-order predictor-corrector method. The integration step used in the batch simulation is 1/2000 seconds. As indicated in figure 6, an integration step of 1/1600 seconds results in a small degradation in accuracy with significant degradation occurring for larger steps. In addition to the actuator, turbulence, aeroelastic, and filter dynamics, the batch simulation also simulates the digital controller. The effects of computational delay and quantization are modeled.

Pre-Test Hot-Bench Implementation

The pre-test hot-bench simulation was implemented in a manner very similar to the batch simulation. State vectors (and associated state derivative vectors) that included all but

the anti-aliasing filter states were formed. The vector of state derivatives was integrated numerically with no assumptions of linearity. Once current-time flexible mode accelerations were calculated from the current-time positions and velocities, an Adams-Bashforth second-order (AB2) predictor method was used to predict the velocities at the next time step. These predicted velocities were then used in a trapezoidal integration scheme to generate predicted flexible mode positions. To wit:

$$\begin{aligned}\dot{\eta}(t+h) &= \dot{\eta}(t) + \frac{1}{2} (3 \ddot{\eta}(t) - \ddot{\eta}(t-h)) \\ \eta(t+h) &= \eta(t) + \frac{1}{2} (\dot{\eta}(t+h) + \dot{\eta}(t))\end{aligned}$$

Using this modified AB2-based method, accuracy comparable to the Runge-Kutta second-order predictor-corrector formula (RK2) used in the batch simulation was achieved with a single pass formula. Note that the RK2 formulation requires two derivative evaluations per time step, whereas AB2 is a single pass formula which gives up some gain accuracy for phase accuracy and is typically used in real-time applications. The integration step of 1/2000 seconds used in the pre-test hot-bench simulation was small enough that the batch and hot-bench simulation results compared favorably.

The anti-aliasing filters were handled separately from the aeroelastic and actuator states. A scalar form of the state transition equations for a constant input over the interval was used. For the single-pole anti-aliasing filters given by,

$$\frac{x(s)}{u(s)} = \frac{1}{(s/\omega_{aa})+1}$$

the constant input state transition equation is

$$x(t+h) = e^{-(\omega_{aa}h)}x(t) + (1-e^{-(\omega_{aa}h)})u(t)$$

As seen in equations (3), calculating the accelerations of the elastic modes requires solving a matrix equation involving the mass matrix. If the mass matrix is constant, an inversion can be performed prior to a run (in a "reset" mode) and the

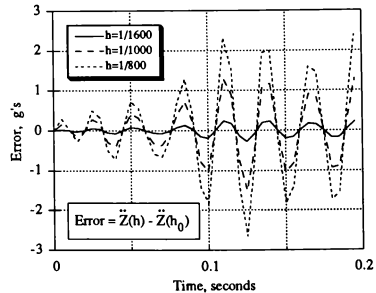
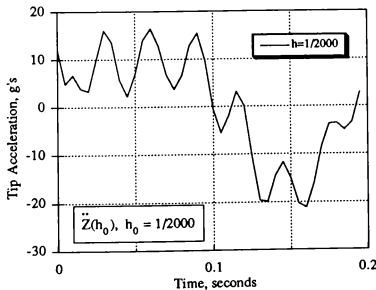


Figure 6 - Effect of step sizes on batch simulation response.

results simply stored for use as the integration proceeds. The mass matrix for the flexible modes, $[M^f]$, (see equation (1)) is both constant and diagonal, and its inversion is trivial. The s-plane formulation used for the unsteady aerodynamic states, however, augments the mass matrix with a fully populated matrix, $[A_2^{ff}]$, that is a function of Mach number. The total mass matrix becomes

$$\begin{aligned} [M] &= [M^f] + \rho \left(\frac{c^2}{8} \right) [A_2^{ff}] \\ &= ([I] + \rho \left(\frac{c^2}{8} \right) [A_2^{ff}] [M^f]^{-1}) [M^f] \quad (5) \\ &= ([I] + [\Delta]) [M^f] \end{aligned}$$

If the density of the test medium, ρ , is to be varied from the simulation control console without losing time synchronization, then the mass matrix equation must be solved at each derivative evaluation. Because the elements of $[\Delta]$ in equation (5) were much smaller than unity, the following approximation was successfully used for the inverse of the mass matrix,

$$[M]^{-1} \approx [M^f]^{-1} \left([I] - \rho \left(\frac{c^2}{8} \right) [A_2^{ff}] [M^f]^{-1} \right)$$

When checked against the exact answer for maximum anticipated values for ρ , induced errors for the pre-test model were less than the precision available in the analog/digital converters used in the simulation loop.

Prior to the November 1989 test, an integration step size of 1/2000 seconds and a compute frame of 1/80 seconds were required for the hot-bench simulation. A compute frame of 1/80 seconds was achieved only if the AFW simulation was the only job on a Cyber 175. Thus, at best, the hot-bench simulation ran 25 (2000/80) times slower than real time, i.e., at a time scale of 1:25. A 1:25 time scale, however, proved to be burdensome while testing the controller performance evaluation mode. Sine sweeps taking 20 seconds in the tunnel would take over 8 minutes in the hot-bench lab. Modifications to the hot-bench simulation implementation, discussed in the next section, allowed the simulation to use a 1/400 second integration step, allowing a 5 fold improvement in time scale.

Post-Test Hot-Bench Implementation

The post-test hot-bench simulation implementation was driven by the need to reduce the time-scale factor to something closer to real time together with the need to accommodate the larger post-test models. If the hot-bench simulation is restricted to a fixed tunnel operating point for a given run, i.e., density, Mach, and airspeed are held fixed, then once the rate limiting is performed on the actuator transfer functions, the remaining dynamics in the simulation are linear. By utilizing a state transition method of discretization on these dynamics, the hot-bench integration step has been increased by a factor of 5, from 1/2000 to 1/400 seconds.

The post-test implementation method, wherein the hot-bench simulation is updated by data extracted from the batch simulation, is depicted schematically in figure 7. The second-order part of the third-order actuator models can be lumped with the remaining linear dynamics. The box in figure 7, labeled "Aeroelastic, Hinge Moment, and Load Equations," represents the state equations of table 1 together with algebraic output equations to estimate the required accelerometer outputs, strain gage outputs, and pressure transducer outputs from the states. Together with the 16 states associated with the second-order part of 8 actuator transfer functions and N (57 pre-test, 156 post-test) states from the aeroelastic model, a coupled linear system of N+16 states, 10 inputs (8 actuator and 2 noise), and 40 outputs can be extracted from the "linear" portion of the batch simulation. To implement the pre-test math model using the post-test state transition method required no model reduction on the extracted model, i.e., the simulation calculations could be completed in the same 1/80 second real-time clock frame used by the pre-test simulation, resulting in a time scale of 1:5. To maintain a 1:5 time scale ratio while implementing the large post-test math model will require model reduction techniques to be applied to the extracted model. Reduction methods utilizing internal balancing techniques are being investigated. Once the model has been reduced, the state transition model based on an integration step of 1/400 seconds is calculated.

The nonlinear portion involves only 8 states, one from each actuator. Each state is integrated numerically with an integration step of 1/1600 seconds. Four integration steps are made to predict the value of the input to the coupled linear system at time $(k+1)h$ where $h = 1/400$ seconds. Since input to the coupled linear system at time $(k+1)h$ is now available, a trapezoidal state transition scheme can be employed. Let $\{u_k\}$ denote the quantity $\{u(kh)\}$. Given the linear dynamic system

$$\dot{\mathbf{x}} = [A]\mathbf{x} + [B]\{u\}$$

if the ramp input signal,

$$\{u(t)\} = \{u_k\} + (t-kh) \frac{\{u_{k+1}\} - \{u_k\}}{h}$$

is defined over the interval

$$kh \leq t < (k+1)h$$

then the following exact solution for $\{x\}$ at time $t = (k+1)h$ exists

$$\{x_{k+1}\} = [F]\{x_k\} + [G_0]\{u_k\} + [G_1]\{u_{k+1}\}$$

where,

$$[F] = e^{[A]h}$$

$$[G_0] = (e^{[A]h} [A]h^{-1} - [A]h^{-1}) [-A]^{-1} [B] \quad (6)$$

$$[G_1] = ([I] - e^{[A]h} [A]h^{-1} + [A]h^{-1}) [-A]^{-1} [B] \quad (7)$$

Note that

$$[G] = [G_0] + [G_1] = ([I] - e^{[A]h}) [-A]^{-1} [B]$$

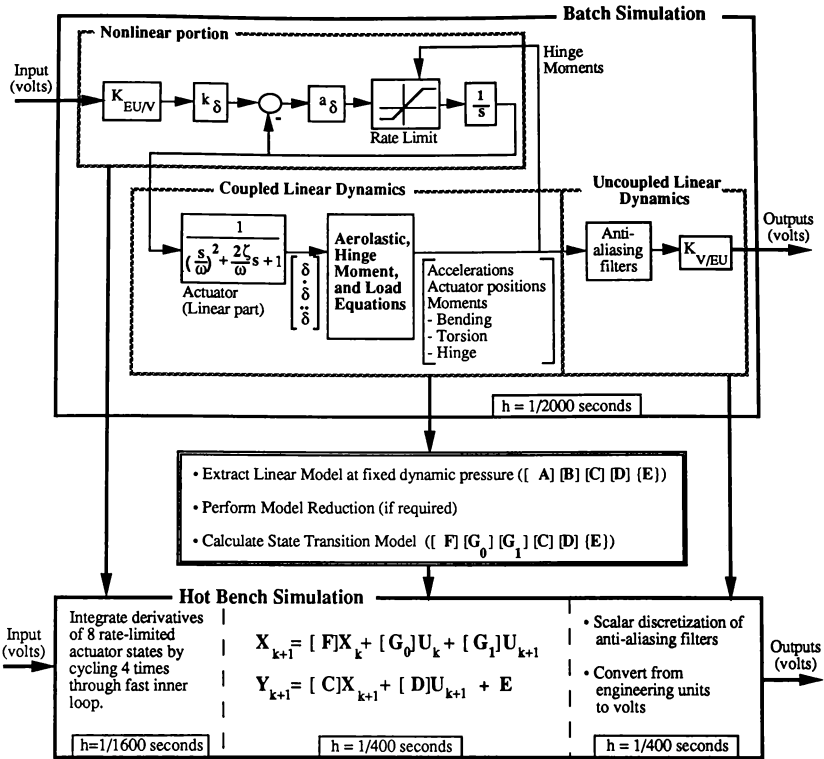


Figure 7. - Data flow from the batch to the hot-bench simulation.

which corresponds to the result one expects to see for $[G]$ if $u(t)$ is assumed to be constant over the interval

$$kh \leq t < (k+1)h$$

Clearly the direct evaluation of $[G_0]$ and $[G_1]$ using equations (6) and (7) will not work if $[A]$ is singular, as would occur if $[A]$ included rigid body modes with zero eigenvalues. However, using the Taylor series expansion

$$\begin{aligned} e^{[A]h} &= [I] + [A]h + \frac{1}{2}([A]h)^2 + \dots \\ &= \sum_{p=0}^{\infty} \frac{1}{p!} ([A]h)^p \end{aligned}$$

and recognizing that

$$([I]h)e^{[A]h} = e^{[A]h}[A]h$$

the equations for $[G_0]$ and $[G_1]$ can be put into a form that can be calculated if $[A]$ has zero eigenvalues. Thus,

$$\begin{aligned} [G_0] &= \left(\sum_{p=2}^{\infty} \frac{1}{p!} (-1)^p ([A]h)^{p-2} \right) h e^{[A]h} [B] \\ [G_1] &= \left(\sum_{p=2}^{\infty} \frac{1}{p!} ([A]h)^{p-2} \right) h [B] \end{aligned} \quad (8)$$

The matrices $[G_0]$ and $[G_1]$ can be calculated by summing the above series until the next term is under some tolerance. When applied to the pre-test model, procedures to sum the series defined by equations (8) converged without difficulty.

The anti-aliasing filters are applied individually to each output signal, which results in a diagonal system. The anti-aliasing filters are therefore not lumped with the coupled linear system to avoid making full matrix operations. The anti-aliasing filter dynamics are digitized in a sequential manner utilizing a scalar form of the trapezoidal state transition method described above. For single pole anti-aliasing filters given by

$$\frac{x(s)}{u(s)} = \frac{1}{(s/\omega_{aa}) + 1}$$

the state transition equations are

$$x(t+h) = e^{-(\omega_{aa}h)}x(t) + g_0u(t) + g_1u(t+h)$$

where

$$g_0 = -e^{-(\omega_{aa}h)}(\omega_{aa}h)^{-1} + (\omega_{aa}h)^{-1} - e^{-(\omega_{aa}h)} \quad (9)$$

$$g_1 = 1 + e^{-(\omega_{aa}h)}(\omega_{aa}h)^{-1} - (\omega_{aa}h)^{-1} \quad (10)$$

Note that the term $(-[A]^{-1}[B])$ in (6) and (7) becomes unity in equations (9) and (10).

Verification

The hot-bench simulation was verified by comparing time history results with the batch simulation. Trajectories were found to match within the width of plotted lines. The open-loop plant dynamics of the batch simulation were verified by comparison with ISAC generated linear models. For each symmetry, a linear model was extracted from the batch simulation using finite differencing. A batch-generated symmetric model and an ISAC-generated symmetric model had different numbers of states because of the way the actuators were handled. The batch-derived model had dynamics for eight right and left control surfaces and the ISAC model had dynamics for only four symmetric control deflections. The corresponding batch and ISAC linear models were compared by overlaying the gain and phase frequency response of each input/output pair. The agreement between ISAC-generated and batch-simulation-derived frequency responses was excellent.

Concluding Remarks

Two simulations, one batch and one real-time, of an aeroelastically-scaled wind-tunnel model were described. The batch simulation was used to generate and verify the real-time simulation and to test candidate control laws prior to implementation on the control computer. The real-time simulation supported hot-bench testing of a digital controller developed to actively control the elastic deformation of the wind tunnel model. Time scaling required for hot-bench testing was discussed. Substantial improvement in the time scale ratio was achieved by application of state transition methods.

References

1. Noll, T.; et al.: *Aeroservoelastic Wind-Tunnel Investigations Using the Active Flexible Wing Model - Status and Recent Accomplishments*. NASA TM-101570 and AIAA Paper 89-1168. Presented at the AIAA/ASME/ASCE/AHS 30th Structures, Structural Dynamics, and Materials (SDM) Conference in Mobile, Alabama, April 1989.
2. Perry, B.; Mukhopadhyay, V.; Hoadley, S.; Cole, S.; Buttrill, C.; and Houck, J.: *Digital-Flutter-Suppression System Investigations for the Active Flexible Wing Wind-Tunnel Model*. NASA TM-102618, AIAA Paper 90-1074. Presented at the AIAA/ASME/ASCE/AHS 31st Structures, Structural Dynamics, and Materials (SDM) Conference in Long Beach, CA, April 1990.
3. Cole, S.; Perry, B.; and Miller, G.: *An Overview of the Active Flexible Wing Program*. Presented at the Fourth Workshop on Computational Control of Flexible Aerospace Systems, Williamsburg, VA, July 11-13, 1990.
4. Miller, G.: *Active Flexible Wing (AFW) Technology*. NA-87-151SL, 1987.
5. Crawford, D.; Cleveland, J.; and Staib, R.: *The Langley Advanced Real-Time Simulation (ARTS) System Status Report*. AIAA-88-4595-CP, September 1988.
6. Hoadley, S.; Buttrill, C.; McGraw, S. and Houck, J.: *Development, Simulation Validation, and Wind-Tunnel Testing of a Digital Controller System for Flutter Suppression*. Presented at the Fourth Workshop on Computational Control of Flexible Aerospace Systems, Williamsburg, VA, July 11-13, 1990.
7. Pototzky, T.; Wieseman, C.; Hoadley, S. and Mukhopadhyay, V.: *Development and Testing of Methodology for Evaluating the Performance of Multi-Input/Multi-Output Digital Control Systems*. AIAA Paper 90-3501. Presented at the AIAA Guidance, Navigation and Control Conference in Portland, Oregon, August 20-22, 1990.
8. Peele, E. and Adams, W.: *A Digital Program for Calculating the Interaction Between Flexible Structures, Unsteady Aerodynamics, and Active Controls*. NASA TM-80040, January 1979.
9. Hoblit, F.: *Gust Loads on Aircraft: Concepts and Applications*, AIAA Education Series, American Institute of Aeronautics and Astronautics, Inc., Washington, D.C., 1988.
10. Tiffany, S.; and Adams, W.: *Nonlinear Programming Extensions to Rational Function Approximation Methods for Unsteady Aerodynamic Forces Which Allow Variable Selection of Constraints*, NASA TP-2776, May 1988.
11. Tiffany, S.; and Karpel, M.: *Aeroservoelastic Modeling and Applications Using Minimum-State Approximations of the Unsteady Aerodynamics*. NASA TM-101574 and AIAA Paper 89-1188. Presented at the AIAA/ASME/ASCE/AHS 30th Structures, Structural Dynamics, and Materials (SDM) Conference in Mobile, AL, April 1989.

SOFTWARE RELIABILITY FOR FLIGHT CREW TRAINING SIMULATORS

George E. Stark
The MITRE Corporation
Houston, TX

Abstract

Flight crew simulator failures are costly and may impact the timing or efficiency of a mission; thus, reliability is one of the most important issues facing simulator developers today. The reliability of a simulator is the probability that a training session of length t can be completed without a failure. This paper defines simulator failure and then identifies and compares the three sources of simulator failure: hardware, software, and human, focusing on the cost of software failure. The paper next describes a model for software reliability measurement and proposes a method for establishing a software reliability object. Data from the NASA Shuttle Mission Training Facility illustrate the technique. Finally, the paper examines the implications of using the method on the software testing successes.

Introduction

Over the years flight crew training simulators have grown in cost and complexity as have the real world systems they model. The modern simulator is a large-scale real-time man-in-the-loop computer system designed to provide the flight crew with a safe, realistic environment in which to exercise procedures related to normal and anomalous conditions. The flight crew simulator depicted in Figure 1 is a typical arrangement of a simulation host computer with distributed nodes which provide visual scenes, motion to the crew station, instructor capabilities, and operator controls.

Flight crew training simulators often require integration with both actual vehicle avionics hardware and software models of supporting avionics systems to reproduce real-world scenarios. In general, software models are used instead of the actual hardware if a set of interconnecting real-world systems can be simplified into a single model, or if the actual hardware is too expensive (in terms of dollars or computational resources).

Simulator software involves time-critical operations synchronized to external events and processes. It is expected to have a predictable response to inputs from a pilot, to malfunctions inserted by an instructor, and to inputs from external interfaces. The software models (outlined in the simulation host shown in Figure 1) are programs that generate realistic outputs from the inputs provided. These programs are usually subdivided into tasks that contain individual entry/exit points so that the execution frequency can be scheduled in an order that ensures minimum calculation latency. Tasks must be completed at predetermined intervals so that time-critical input/output may be serviced with accurate and complete data. A real-time operating system is required for responding to hardware interrupts and maintaining control over the various software tasks. Clearly and completely specifying the interfaces, timing, and scheduling

of the individual programs and tasks is necessary for the success of the simulator implementation.

Specification of simulator software requirements involves identifying the relationships that should exist between the inputs from the external events/internal processes and the outputs to the crew and instructors. These relationships may be explicit or implicit in a specification. An example of a requirement is that no response to crew input should take more than 150 milliseconds longer than the actual vehicle response lag¹. A more complicated example is a visual display that accepts input from several sources and produces a detailed visual scene with ten moving objects, texturing, and occlusion.

Definition of Failure

A simulator fails whenever the produced outputs do not satisfy the input/output relationships specified in the requirements. Three sources of failure exist in any simulator: hardware (HW), software (SW), and human. Table 1, adopted from Shoeman,² displays comparative failure mechanisms for each of these failure sources. This table indicates that all three sources have different failure mechanisms, but that each does contribute to the system failure rate. Since all three components are required to work for a successful training session, the probability of a successful training session of length t (i.e., the reliability of a simulator) can be calculated as the product of the three reliabilities, or

$$R(\text{sim}) = R(\text{HW}) \cdot R(\text{SW}) \cdot R(\text{human}) \quad (1)$$

Software developers assume that the computer instructions are reloaded on a stable medium and that the processor will always execute a given machine instruction exactly as stated in the processor specifications. Whenever these assumptions are violated, the software developer claims the hardware has failed. Furthermore, software developers assume the operations personnel always follow correct procedures in loading the software and that instructors use correct command sequences while interacting with the software. Whenever these assumptions are violated, a human failure is said to occur.

The input and output spaces for flight simulator software are usually so large and the required relationships between them so complex that, even applying the best software engineering technique, situations will arise in which the requirements are not met. Whenever this happens, a software failure³ is said to occur. The cause of a failure is termed a *fault*,³ and it is the responsibility of the software engineer to eliminate it by substituting correct instructions or produce data for erroneous ones or by providing missing data or instructions. The *reliability*³ of the software is

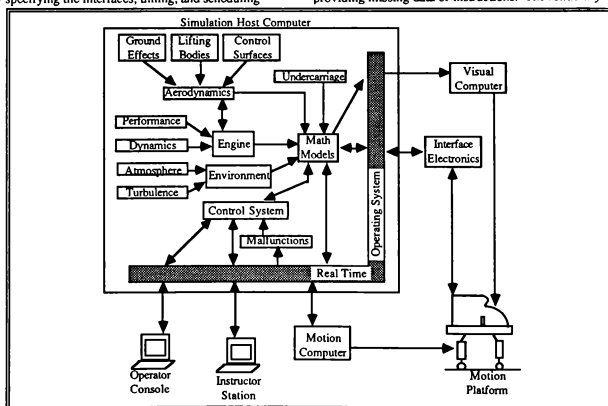


Fig. 1. A Typical Crew Training Simulator Architecture.

Failure Mode	Hardware (HW)	Software (SW)	Human
Fabrication	Bad solder joints Bad component installed Mechanical misalignment	Typographical error Wrong version subroutine Incompatible operating system	Wrong disk mounted Wrong switch position set Reload button pushed in error
Design Error	Component rating too low Metal parts exposed to corrosion An address load clears the accumulator	Incorrect re-initialization Interchanged branches Series expansion does not converge for all values	Enter wrong data in response to a request Illogical key data sequence entry Operator follows incorrect user guide steps and clears memory
Component Overload	Capacitor with 50 V rating is used in circuit with 100 V transients HW cannot keep up with 1200 baud input even though the specs call for operation at that rate	Control system designed to handle 100 targets, drops targets without warning when more than 100 enter the control zone Input module of a text editor cannot keep up with a fast typist	Operator cannot handle more than 50 targets without overloading vigilance capacity An operator forgets the proper command sequence because there are too many steps
Wear Out	A mechanical clutch slips after 5000 hours Insulation cracks on wires after 10 yrs causing shorts	No analogous effect	Errors due to cumulative fatigue

Table 1. Comparative Failure Mechanisms - Hardware, Software, and Human.

thus defined as the probability of failure-free operation for a specified time interval in a specified environment.

Cost of Software Failure

Training simulator software failures result in a directly quantifiable cost as well as a qualitative cost. The directly quantifiable cost has three components: (1) the cost of lost training time, (2) the cost of restoration, and (3) the cost to repair the fault in the software. The lost training time cost includes not only the cost associated with failure isolation and equipment downtime but also the cost of the instructor and the crew time. Restoration cost includes the cost of the time it takes to reinitialize the simulator and return it to a point at which the training session can continue. For example, assume a simulation fails two hours into a simulated mission, after restoring the simulator to operational, the simulator must be flown to the point at which the failure occurred to continue the training session. The repair cost is generally an off-line cost, since the repair is usually completed by a staff of programmers at a remote location and may be handled through a separate discrepancy reporting process that has a time lag associated with it.

In the case of the Space Shuttle Mission Training Facility (SMTF), located at the NASA Johnson Space Center, the estimated cost of running a training session is \$6450 per hour⁴. During the first quarter of 1986, the downtime associated with a critical⁵ software failure in the facility averaged 1.21 hours, including the restoration time, which averaged 10 minutes⁶. Actual data for the SMTF was not available. For the sake of illustration it is assumed an average of five programmer hours for problem analysis, at a rate of \$50 per hour, and five programmer on-line coding hours (with terminal access to a development facility), costing an average of \$75 per hour, were required to repair each fault. Thus, the total direct cost of a critical software failure would be \$8410.

In addition to the quantifiable costs of a software failure, there are the qualitative costs related to the timing of future missions and to negative training. For example, if a crew member must complete a specified number of simulator hours to gain certification in some mission maneuver, the reliability of the simulator impacts that member's ability to accumulate the required hours. This may cause the mission to be postponed. If the mission is executed on schedule, the inability to complete training increases the risk of the maneuver during the mission. The impact of negative training on the crew members, although difficult to measure, is perhaps the most important cost. A software failure that repeatedly disrupts the flow of the task being trained may cause the student to incorrectly execute a maneuver in the real vehicle, thereby jeopardizing the success of the mission.

Given these costs, it becomes apparent that software reliability should be specified during procurement of training simulators. If the software performs below the specifications, the developing contractor

should be assessed a penalty. The penalty should be proportional to the schedule impact experienced by the user due to the rework required to meet the requirement. Further, the reliability should be measured during training operations so that corrective action can be taken if changes to the simulator cause the reliability to drop below the required level.

The following sections describe a method for measuring the reliability of simulator software and propose a method for establishing a reliability objective based on minimizing the costs of the software throughout the operational life of the simulator. Finally, impacts on the developer's testing process are discussed.

Software Reliability Measurement

The inputs to be processed by the simulator software arrive in random order, at random times. In practice an operational piece of software operates successfully for the vast majority of its inputs. It is only under a specific set of input conditions that some inherent fault manifests itself as a failure. Therefore, software failure occurrence can be considered a random process. Furthermore, even if the software is perfect, there is no way to prove this with certainty on large systems. Proving software correct has been demonstrated on small programs, but is virtually impossible on large systems⁷. The best that can be done is to establish (with an appropriate level of confidence) that the probability of failure is small compared to the consequences of such a failure. That is, a higher probability of failure is acceptable if a software failure results in a small monetary loss than if the failure results in the loss of an expensive spacecraft or human life. In either case, however, the probability of failure must be verified.

Over the past twenty years, more than forty models have been proposed for estimating the reliability of large software systems^{8,9}. Among these, the non-homogeneous Poisson process (NHPP) model is described in this paper. The NHPP was chosen for three reasons: simplicity, applicability, and tool availability. The model is simple in concept; an extensive mathematical background is not required to understand the nature of the model or its assumptions; its parameters have readily understood physical interpretations; and the data required for the model is simple to collect. The model is applicable to a wide variety of software development efforts including real-time flight simulators. It is a reliability growth model. This means that as the test/debugging phases of software development disorder and correct design flaws, the probability of failure decreases. Finally, while these attributes apply to several models, the NHPP was a convenient choice because a tool is available to calculate the model parameters and performance measures⁹.

* SMTF failures are classified as "critical", "degraded", and "no lost time". A critical failure is a failure that results in a hard stop of the simulation or requires that the simulator be brought down and repaired prior to continuation of the training session.

The assumptions behind the NHPP model are as follows:

- The total number of faults to be found and corrected in the software is a Poisson distributed random variable that approaches a constant (say, N) if debugging is carried out indefinitely.
- Tests represent the environment in which the simulator will be used.
- Each failure is caused by one fault. The faults count detected during non-overlapping time intervals are independent of one another. This assumption allows the words failure and fault to be used interchangeably during the rest of the discussion.
- The expected number of failures in a small time interval is proportional to the length of the time interval multiplied by the number of faults in the software at the beginning of the time interval.

Using assumptions (a)-(d), the expected number of software faults detected and corrected by some time T, $f(T)$, is given by

$$f(T) = N[1 - \exp(-\lambda T)] \quad (2)$$

where N is as described in (a), and λ is the proportionality constant from (d). λ takes on the units failures/time, and thus can be interpreted as the rate at which faults are removed from the system (i.e., fault reduction factor) in the reliability growth model.

Furthermore, the probability of experiencing a total of x software failures by time T is given by

$$\Pr(\# \text{ failures by } T = x) = [\exp(-f(T)) f(T)^x] / x! \quad (3)$$

for $x = 0, 1, 2, \dots$. Finally, the probability that the software will be operational for at least y time units, given that the last software failure occurred at some time $T = z$, is the reliability function, $R(y)$, or

$$R(y) = \Pr(Y \geq y | T = z) = \exp(-f(y) \exp(-\lambda z)) \quad (4)$$

Three methods exist for estimating the parameters N and λ in equation (2) given individual times of error occurrences: Method of Moments, Least Squares, and Maximum Likelihood (MLE). Each method is discussed by Shooman², but since MLEs have the best statistical properties, Goel and Okumoto¹⁹ have derived these estimators for N and λ by letting z_i represent the failure time of the i th error, ($i = 1, \dots, b$). The MLEs are then the solutions to the following system of equations:

$$N = b / [1 - \exp(-\lambda z_b)] \quad (5)$$

and

$$\lambda = [\theta + (N/b)z_b \exp(-\lambda z_b)]^{-1} \quad (6)$$

where b is the total number of errors detected, and θ is the average of the failure times, that is, $\theta = \Sigma(z_i/b)$, $i=1, \dots, b$.

Establishing a Software Reliability Objective

Using the previous formulation, Goel and Okumoto have proposed a method for determining when to release a software product from testing¹⁹. Additionally, these equations allow management to assess the current status of a software project throughout the testing and operations phases of a project by trading off reliability with the costs associated with testing and operational failures. This trade can answer the two questions: (1) What reliability objective will minimize the simulator's life-cycle cost? and (2) How much testing is required to ensure this objective is met?

A reliability objective is normally established during the requirements definition or project planning phases of the system development effort. Unfortunately, unless a project manager has previously collected software reliability data on a similar project, it is difficult to initialize the model. This is a classic chicken-and-egg problem. Failure data is required to initialize the model and generate requirements; however, without requirements there is no justification to gather the data. The way out of

this dilemma is to estimate values for the required parameters using data from other projects or accepted rules-of-thumb.

Table 2 contains estimates of N and λ for four projects, along with the size of the project (in source lines of code), and the number of errors recorded during system test. The data are from various sources^{1,1,2,3,14}, and while only project C is a flight simulator, projects B and D are avionics systems. Project A is a military command and control system. Note that the data from project C was collected during operations and, hence, may not accurately reflect testing-phase parameter estimates, since much of the software was very mature.

An alternative to estimating values based on Table 2 is to use the rule of thumb suggested by Hecht¹⁵. That is, N equals 2% of the source lines of code and lambda is initialized to 0.10. These appear to be conservative planning estimates based on the data in Table 2.

Data Set	N	λ	Size (KSLOC)	failure sample	test time (days)
A	130	.042	180	101	364
B	45	.460	120	40	unknown
C	32	.004	1,100	24	58
D	1348	.124	unknown	1138	100

Table 2. NHPP Parameter Values for Four Data Sets.

Attempts to predict software reliability prior to the start of integration testing are still considered preliminary^{16,17}. Consequently, any costs incurred in the development phase prior to testing are considered fixed and are not changeable by altering the required reliability. Furthermore the software is considered released at the end of testing and the release date denotes the beginning of operations.

Given these considerations, let

- C_1 = cost of fixing a fault during the test phase
- C_2 = cost of a failure during operations
- C_3 = cost of testing per unit time
- t = expected software operational lifetime
- T = software execution time
- $f(T)$ = expected number of faults detected and corrected by time T

With these variables, the expected cost of fixing faults during test is $C_1 f(T)$, and the expected cost of failures during operations is $C_2 (f(t) - f(T))$. The cost of testing is $C_3 T$. So, the expected life-cycle cost (LCC) at any execution time is the sum of the three components, or

$$LCC(T) = C_1 f(T) + C_2 (f(t) - f(T)) + C_3 T. \quad (7)$$

Substituting equation (2) for $f(T)$ and differentiating with respect to T, yields a cost minimum at

$$T = \ln [N\lambda(C_2 - C_1)/C_3] / \lambda. \quad (8)$$

This methodology is illustrated using cost data reported on the SMTF and making reasonable assumptions where data was not available. The values used in this example are shown in Table 3. Substituting these values into equations (7) and (8) resulted in Figures 2 through 4.

Size (KSLOCs)	500
N	250
λ	0.125
cost of fixing a fault during test	\$2400/hr
cost of an operational failure	\$8410/hr
cost of testing	\$4495/hr

Table 3. Parameter Estimates Used for Establishing a Software Reliability Objective.

Figure 2 shows the curves associated with each component of the life-cycle cost equation. The cost of failures during operations decreases quickly as testing execution time increases. It becomes zero when all faults are removed from the system. This cost decrease in operations is

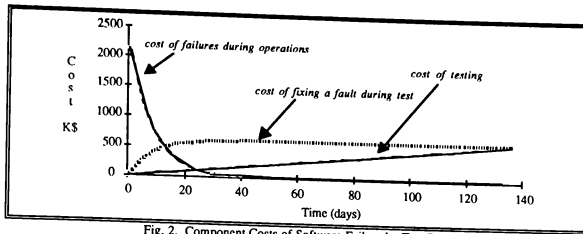


Fig. 2. Component Costs of Software Failure by Test Days.

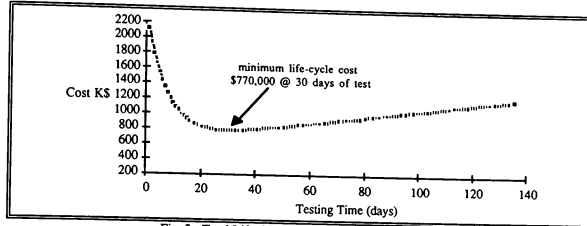


Fig. 3. Total Life-Cycle Cost by Testing Time (Days).

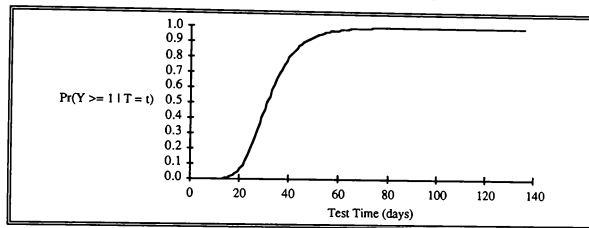


Fig. 4. Expected Reliability vs Test Time (Days).

offset, however, by an increase in the cost of debugging during test and the linear increase in the cost to test. The cost of fixing a failure during test reaches its maximum at N^*C_i , when all faults have been detected and corrected. The cost of testing continues to rise even if there are no more faults in the code, since testers, trying to break the simulator, are still using system resources at a constant rate.

From Figure 3, if the software is released to operations with no testing at all, the total cost will be approximately \$2.1 million dollars, and (from Figure 4) the probability that the software operates without failure for 1 day is zero. By combining data illustrated in Figures 3 and 4, the solution that minimizes life-cycle cost is $R(1 \text{ day}) = 0.50$, at a total cost of \$766,000. It is this reliability value that should be specified in the requirements if the customer is interested in the probability of successfully completing one training session of length one day ($y=1$ in equation (4)). Notice also from Figure 3 that the cost penalty for over-testing is not as severe as for under-testing. For example, under-testing by ten days costs \$43,150 more than the optimum; whereas over-testing by ten days costs only \$19,600 more. This indicates that software managers should test longer, when possible, to minimize the risk of not meeting the objective due to the variance associated with parameter estimation. The software should continue to be debugged after release, resulting in a higher reliability product in the future.

Effects of Software Reliability Measurement on the Testing Process

After an objective is established and agreed upon by the developer and the user, verification that the simulator meets that level of reliability

must occur. This verification takes place during the software testing process. Software testing is normally conducted in three stages:

- (1) *Unit testing* is executed by the development programmer on his/her particular subsystem. It is traditionally done in a development facility using software drivers to generate inputs to the subsystem rather than in an operational environment with the actual hardware or other software subsystems providing the inputs. Faults discovered during this stage are generally not recorded or tracked.
- (2) *Integration testing* is performed by a team of development programmers and test engineers. During this stage subsystems are combined through a step-by-step build up of the simulator. As it is integrated, each newly-added subsystem is tested in conjunction with the simulator hardware and previously-combined subsystems. Faults are usually recorded and tracked during this stage, but the requirements for using current software reliability measurement models (i.e., tests are not executed in a manner similar to the operational environment using a complete system) are not met by the nature of the integration.
- (3) *System testing* is executed by the development test team (sometimes in conjunction with the customer) to verify that the simulator meets the functional requirements specification. Faults are recorded and tracked. The testing is generally done by flying the simulator in its intended operational environment. Thus, it is during this phase of testing that current software reliability models have the most applicability.

Using software reliability measurement requires two important modifications to the system testing effort. The first is a change in the collection of data during testing. The simulator execution time between failures, rather than the number of successful tests, is required to estimate the model parameters. This data can be measured directly with a hardware monitor attached to the processors running the simulation or by a software monitor embedded in the code. When neither of these options is available, the time can be estimated based on the number of interrupts handled during some known cpu-intensive time period or by using a discrete-event simulation of the system. Second, test cases should be selected at random from the input space in accordance with a specified training scenario. That is, instead of concentrating testing on one function at a time, the cases should be selected at random from all of the functions; the most used functions being assigned higher frequencies of execution during testing.

Summary

This paper has provided an overview of the major steps required to establish a software reliability program for training flight simulation software. The process starts with the definition of software failure. Given the definition, the paper describes a method for determining the direct cost of software failures. Actual failure cost data from the Shuttle Mission Training Facility was presented as an example. The next step in the process is to choose and initialize a model to specify the reliability objective of the software. The NHPP model was described and two methods for initializing it were presented. Given the specified objective, the direct cost of operating the simulation software and the amount of system testing required to deliver the simulator can be estimated. The final step is to verify that the objective has been met. This was described in the paper along with an overview of the measurement requirements and their impact upon the testing philosophy.

The cost of software failures, both direct and indirect, is the largest cost of operating a training flight simulator today. This makes specification and verification of software reliability a primary concern of simulation software users. Contracts for flight simulators should specify a reliability objective that minimizes the life-cycle cost of the software, unless safety concerns drive the user to a more expensive, higher reliability requirement.

Acknowledgements

This paper was patterned after a talk presented by Dr. A. Frank Ackerman at the AIAA Space Based Observation Systems Committee On Standards in November 1989. Further thanks are due Ankur Hajare, Tim Featherston, and the other MITRE employees for their insightful comments on earlier drafts of this paper.

References

1. Federal Aviation Administration, "Airplane Simulator and Visual System Evaluation", Advisory Circular #120-40A, July 1986.
2. Shooman, M. L., *Software Engineering: Design Reliability Management*, McGraw-Hill, Inc., 1983.
3. ANSI/IEEE Std 729-1983 in *Software Engineering Standards, Third Edition*, "Glossary of Software Engineering Terminology", IEEE, 1989, pp 1-38.
4. Beaty, W. K., personal communication, The MITRE Corporation, April 1990.
5. Stark, G. E., "Dependability Evaluation of Integrated Hardware/Software Systems", IEEE Trans. on Reliability, vol. R-36, October 1987, pp 440-444.
6. Farmer, W. M., Johnson, D. M., and Thayer, F. J., "Towards a Discipline for Developing Verified Software", MTP-261, The MITRE Corporation, August 1986.
7. Farr, W. F., "A Survey of Software Reliability Modeling and Estimation", Naval Surface Weapons Center, NSWC-TR 82-171, September 1983.
8. Stark, G. E., "Software Reliability and Its Application to Future NASA Projects: A Review of the Literature", WP 6301, The MITRE Corporation, June 1986.
9. Vienneau, R., "User's Guide to a Computerized Implementation of the Goel-Okumoto Non-homogeneous Poisson Process Software Reliability Model", IIT Research Institute, Data Analysis Center for Software (DACS) report 90-0027, November 1987.
10. Goel, A. L., and Okumoto, K., "Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures", IEEE Transactions on Reliability, vol. R-28, No. 3, August 1979, pp 206-211.
11. Musa, J. D., "Software Reliability Data", Bell Telephone Laboratories, Data Analysis Center for Software (DACS), January 1980.
12. Schafer, R. E., Alter, J. F., Angus, J. E., and Emoto, S. E., "Validation of Software Reliability Models", RADCO-TR-79-147, June 1979.
13. Stark, G. E., and Shooman, M. L., "A Comparison of Software Reliability Models Based on Field Data," presented at ORSA/TIMS Joint National Conference, Miami, October 1986.
14. Moek, G., "Comparison of Some Software Reliability Models for Simulated and Real Failure Data", fourth IASTED International Symposium and Course "Modelling and Simulation", Lugano, June 1983.
15. Hecht, H., and Hecht, M., "Software Reliability in the System Context", IEEE Trans. on Software Engineering, vol. SE-12, No. 1, January 1986, pp 51-58.
16. Henry, S., Kafura, D., Mayo, K., Yemeni, A., and Wake, S., "A Reliability Model Incorporating Software Quality Factors", Proceedings of the Annual National Joint Conference on Software Quality and Reliability, March 1-3, 1988, pp 340-352.
17. Soistman, E. C., and Ragsdale, K. B., "Impact of Hardware/Software Faults on System Reliability - Study Results", RADCO-TR-85-228, December 1985.

VALIDATION TESTING: THE DRIVER FOR PRODUCT QUALITY

John F. Sarnicola

Senior Staff Scientist
Link Miles Corporation
Binghamton, New York

Abstract

One of the single largest most influential set of constraints affecting the quality of advanced simulation devices is embodied in the FAA regulations enforce during the period of design. The impact of required validation and verification testing is analyzed with an emphasis on tolerances, complexity, fidelity, hardware design and cost. Similarly, changes and improvements by airframe manufacturers, better and more complete data, and, more sophisticated models and modeling techniques, all have varying effects on quality and cost. Significant parameters in the design process are considered leading to the development of a generalized procedure when approaching simulator design for improved pilot training. The analysis can be applied when considering the inclusion of additional validation testing or modifying existing tests and its potential impact on training. Discussions of subjective testing, perception and cuing levels, and, modeling are interwoven in the analysis, providing a basis for decision making.

Introduction

Of the many constraints placed on a designer of advanced simulation equipment, FAA tolerances are a major driver for ensuring fidelity, quality level and consistency. This intern establishes the degree of repeatability and reliability of the training device. Of concern is the tendency for tolerances outlined in advisory circulars to easily become the minimum standard of the simulator manufacturer. Although individual systems are typically designed to exceed minimum standards, the systems are tested to only meet the minimum standards. Hence, there is a natural tendency to gravitate to the minimum if care isn't taken to maintain self checks and internal quality controls. One of the main reasons for this tendency is the economics associated with competition. If not kept in check, these FAA tolerances and guide lines can easily become a "crutch" for the simulator manufacturer when dealing with subjective portions of certification. It is important to keep active, the intent of such regulations, i.e.; that the regulation is the basis for minimum performance of the device or process under consideration. This should not be the only standard or minimum design standard of the simulator manufacturer. For this

reason it is good practice to include specific and detailed statements of work, including tolerances and quality acceptance guidelines as part of purchase agreements.

When investigated, it is often found that a reason for inadequate design, hence, poor performance of a system, is that the designer was unsure or unaware of the intent, extent and/or basic data associated with the system during initial design. The assumptions made by the designer at the initial stages of design, when not based on concrete information has nearly an unrecoverable effect on the long term operational characteristics and over all quality of the system. It is critical that hardware designers be provided with clear and precise statements of work including system requirements, and, that design and manufacturing tolerances be established with all existing applicable regulations considered. Manufacturers standards must strive to exceed FAA requirements using the latest hardware and modeling techniques available, and in a competitive manner. FAA requirements can, in fact, be of valuable assistance to the designer in identifying systems considered to be critical to good training; particularly in the areas of flight controls, aerodynamics and autopilot.

Once a good design has been developed on paper and then prototyped in the laboratory, it is necessary to build the production hardware. The reality of manufacturing and producing hardware to a pre-agreed schedule, however, is often in direct conflict with the ideals of any standards established during design. If not carefully controlled, final qualification can be relegated to "passing FAA tests". During the FAA qualification period it is simply too late in the design-build process to provide major substantive correction for either design features or lax manufacturing and assembly procedures; all of which can have a long lasting operational cost impact. Established manufacturing standards and procedures must be compatible with the intended functionality of the systems. If this is the case, and the standards are strictly enforced, then an acceptable level of product quality should be nearly an automatic result.

Although good hardware design is a primary requisite for a successful operating system, it must be accompanied by complementing software. Like hardware,

software can be designed to varying quality standards. The impact on the resulting system operation can be of equal magnitude of any physical limitations. The combined use of hardware and software must necessarily be a blend of acceptable compromise in any unified design approach. Software modeling cannot compensate for all of the deficiencies in a poor hardware design, and similarly, hardware cannot compensate for all software deficiencies. Hardware and software systems designed in isolation result in systems with less than acceptable operation. It is often required disruptive and substantive modifications in the final stages of manufacture.

One of the primary reasons for recertification and recurrent training being delayed is the lack of performance of the device. This situation is most often traced to hardware degradation. This degradation is a function of both design and, operational and maintenance procedures. It is a measure of the quality of the device or system. Since, a simulator is a composite of many systems and subsystems, whose design is subject to the discretion of a multitude of technical people with varying capabilities, and operated and maintain by yet another group of individuals with varying talents; then it follows that design, manufacturing and fabrication "tolerance bands" must be tighter than those of the certifying agency if a successful simulator is to be produced. It will be shown that the average quality of these systems has a marked effect when attempting to optimize training performance of any device.

Additionally it will be shown that the impact of tightening tolerances results in a significant increase in pilot training cost. Hence, the reality of the nature of manufacturing makes it imperative that FAA tolerances be accurately and intelligently established, and, the impact of any tolerance changes be understood both technically and economically. Although beyond the scope of this analysis, the net benefit of tightening tolerances when comparing cost and improved pilot training must be carefully considered.

Hardware and Software Design

The ease or difficulty with which initial and recurring acceptance of a training device is achieved has its roots in the basic design of both the software and hardware systems. With the tremendous advances in computing research and resulting electronics technology, there is no question that at the present, computing system capability significantly exceeds the ability of the mechanical hardware to respond. Significant amounts of time and money have been spent on upgrading and improving simulator software systems. The shift from analog to digital systems has resulted in computational systems with more capability, flexibility and greater reliability. The emphasis on software, computer systems, and electronics technology has been so great as to dwarf developments in basic mechanical systems design. Research in new, inventive and improved mechanical systems

has fallen behind the software and electronics effort.

Once developed, software is a static element. Hardware, however, does degrade, break or otherwise result in operational and maintenance problems. As a worst case it can be easily demonstrated that even with the most sophisticated software modeling, it is not possible to compensate for deficiencies in basic hardware design and operation. In no case should attempts be made to compensate for hardware deficiencies, in software, beyond original design limits. Occasionally, certain effects can be masked or worked around by software methods, but in general the hardware must be designed and fabricated correctly. It is precisely these deficiencies in hardware operation that account for the bulk of long term operating costs. It is essential that both mechanical and electrical hardware designers have a full grasp of system requirements. Such intimate knowledge can only come from developing a close working relationship with software and software/hardware integration personnel. Without this close tie between disciplines, the 'tweakability' that is required for final system tuning will very often be lacking.

Mechanical and electrical hardware systems must be designed to complement the newer "digital" systems. It is suggested that many here-to-fore accepted standard design practices may require revision to accommodate newer technologies. For example; transducer design and application, instrumentation, drive systems, friction analysis and materials usage, to mention a few. Although beyond the scope of this paper, the concept of "matched" hardware and software design needs to be addressed.

Operational Requirements

Presently, there are two major sets of constraints a simulator manufacturer must design and build to. First, the FAA requirements composed of functional and performance tests, and secondly, additional customer requirements outlining specific features and training plans. Occasionally there is some overlap on items. However, typically a customer has additional or unique requirements for a particular system based on individual circumstances.

Tolerances from any source provide a major obstacle for most manufacturer's of simulator systems for many reasons. First, the quality and availability of airframe data varies considerably between manufacturer's. Good quality data may only be available through aircraft instrumentation and testing due to the age of the aircraft to be simulated. Secondly, there exists no standard format which regulates the quality of data measured directly from aircraft by various groups using differing methods. Although the IATA has developed a document intended as a guide to identify the basic elements necessary to be implemented for adequate fidelity, the issue of accuracy and quality are not covered nor does IATA's

document cover all aircraft systems or avionics types in a consistent manner. As a result, anomalies can occur when data is single sourced. Communications between airframe manufacturers and simulator manufacturer's to resolve such issues is often a necessary and expensive protracted experience. Thirdly, data is extremely expensive. This may force the simulator manufacturer (for non-technical business reasons) to wait to the last possible time before purchasing the necessary data package. As a result, it is entirely possible that systems may be designed around outdated information or based on imprecise assumptions. Depending on the individual circumstance, once a design is complete, it can be extremely difficult and expensive to significantly change. Fourth, the simulator manufacturer is ever increasingly being asked to implement the airframe manufacturer's math models verbatim. Although this results in uniformity of models between simulator manufacturer's, it also limits the ingenuity and inventiveness of the modeler. As a result, simulator manufacturer's tend to become more alike since they must perform the same tasks in nearly the same fashion in order to stay reasonably competitive. This homogenizing effect results in higher costs by limiting other potentially more economical solutions. In addition, at times it is necessary to model systems based on provided theoretical systems computations. Depending on the airframe manufacturer, such models are "documentation after system design". Such efforts can lack real world fidelity. Simulation models however, must model the physical real world including transients and anomalies. Correcting or improving the subjective aspects of operation to meet customer demands is expensive and time consuming.

Quantifying Performance

In order to bring perspective to the interrelation of parameters affecting simulator performance, it is useful to develop an overall mathematical expression which relates the major first order effects. The analysis which follows attempts to relate variables associated with maximizing quality training. The case of maximizing training, i.e.; per pilot recurrent training cost is looked at in terms of the loss in production rate attributable to downtime due to out-of-tolerance systems as the result of design, manufacture and operational quality. It will be shown that the resulting downtime can be very high with training devices having large numbers of complex systems. The following is an economic analysis of the performance of a composite training device and the effects that design quality and mandated tolerances have on the production rate of pilots. It is intended to be applied to recurrent training situations, however other training scenarios can be considered with appropriate modifications.

For this analysis, it will be assumed that the simulator or training device is comprised of a set of systems, Fig. 1, which represent the aircraft. Ideally,

all of these should operate correctly for complete and successful training of each pilot.

$$\text{AIRCRAFT} = \{S_1, S_2, S_3, \dots, S_n\} \quad (1)$$

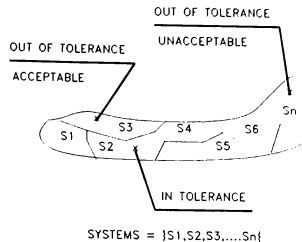


FIG. 1 - SIMULATED AIRCRAFT SYSTEMS

As a basis, this analysis will consider that individual simulator systems can impact training in one of three ways, they can be:

1. In tolerance.
2. Out of tolerance, training can proceed on other systems. Indirect downtime for training that will be required to be made up.
3. Out of tolerance, downtime occurs when all system training must stop.

In order to develop an understanding of the interaction of the many variables it is advantageous to define per system relationships as follows:

$$\text{Total Training Time} = tN \quad (2)$$

where:

t = Training time, hours/pilot.
 N = Number produced, pilots.

The downtime of the training device can be represented by:

$$\text{Downtime} = mnNTQ/E \quad (3)$$

where:

m = Proportion of out-of-tolerance systems causing simulator stoppages, stoppages/out-of-tolerance system.

n = Number of active systems required for a pilot to be exercised on for successful training, systems/pilot.

T = Time required on average to correct an operational problem and restart a system, hours.

Q = Quality level of simulator systems. Ratio, out-of-tolerance systems/in-tolerance systems.

E = Specification Factor: reflects a relative measure of tolerance required to be met. Example: Basis Tolerance E=1; Proposed Tolerance E=.75 (25% tighter).

The proportion of downtime D, on the device can be calculated from:

$$D = \frac{\text{Downtime}}{\text{Training Time} + \text{Downtime}} \quad (4a)$$

$$D = \frac{mnNTQ/E}{Nt + mnNTQ/E} \quad (4b)$$

$$D = \frac{mnQ}{Et/T + mnQ} \quad (4c)$$

Although there are many varied systems on differing aircraft, real values for the indicated unknowns exist or can be calculated from existing data for each training situation. Values will vary between aircraft types and training facilities. Results of equation (4c) are shown in Figs. 2a and 2b for a range of typical values.

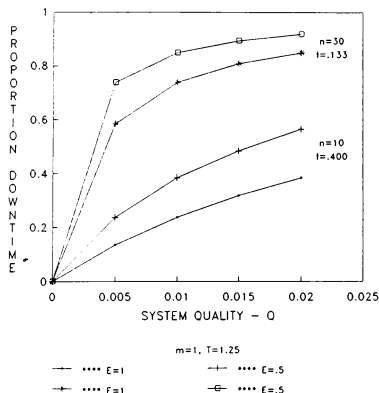


FIG. 2A - PROPORTION DOWNTIME (m=1)

Fig. 2a shows the effect of average quality level on the downtime for an aircraft with 10 and 30 active systems. Fig. 2a assumes that all out of tolerance systems result in downtime (m=1). Fig. 2b represents the same situation except that training downtime is only marginally affected by out of tolerance systems (m=.1). It should be noted that for this analysis all systems are treated equally in terms of quality level. This is justified by considering that each individual system quality level, whether weighted (depending on its critical nature) or not, can be represented as an average. There may be several methods of determining this value depending on available data. For the purposes of this

analysis, however, an overall average is sufficient to demonstrate the magnitude of effects involved.

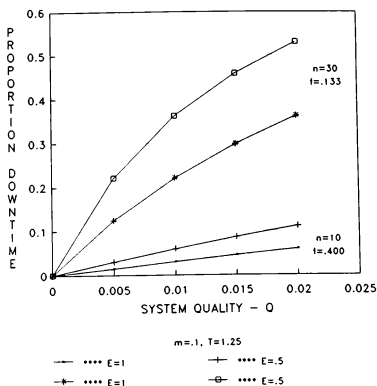


FIG. 2B - PROPORTION DOWNTIME (m=.1)

The simple exercise represented by Figs 2a and 2b demonstrates a familiar phenomena, i.e.; as the quality level of the systems improve (smaller values of Q), there is a corresponding reduction in downtime. The anomaly is that, keeping all other things equal, tightening tolerances (reducing E) has the same general effect as reducing the quality level of the systems. Simply put, with no change in basic design or maintenance and operating methods, it will always be more difficult to obtain, and keep, a reduced tolerance system in acceptable operation.

Quality, Tolerance and Training Time

As anticipated, time dependent variables associated with downtime or production rate show a dramatic improvement as the quality level improves. This will always be the case when no consideration for cost is made. In the real world however, the two cannot be uncoupled. Improved quality in design and operation have their price.

Some revealing basic results can be obtained from Eq. (4c). First, the difference between Figs. 2a and 2b is the value of m; the measure of out of tolerance systems which stop training. A comparison of the two figures for the case considered shows that given the alternative, it is better to have an out of tolerance system that may result in a limited amount of training, than to have a system that stops training altogether. It is noted that the precise value of m is empirically determined from a combination of both technical and training parameters. The technical aspect should account for clear cut hardware operation while the training aspect accounts for the

operational degradation and is a reflection of the point in operation where negative training would start to occur. The cost of this retraining must also be taken into account, and this will be considered later.

It can be observed from the slope of the curves, that improved system quality (design, manufacture and operation) have a large impact on downtime. Higher quality systems are considered those with an index of .01 and smaller. It is interesting to observe that for the conditions selected, the value of m has the same order of magnitude effect on downtime as Q .

As already suggested, the impact on downtime of simulators with a high number of systems (large n) is significant. Given comparable circumstances as represented in Figs. 2a ($m=1$) and 2b ($m=.1$), it can be seen that reducing the number of operating systems has a major effect on the simulator downtime. Hence, an excellent argument for transferring the bulk of training procedures to less expensive Flight Training Devices (FTD's) which are comprised of fewer discrete systems that can be more closely controlled. This also suggests that it is desirable to reduce simulator systems and complexity wherever possible. This does not imply that systems cannot be highly sophisticated, but rather there should be fewer systems of simpler design.

Of equal importance and impact on downtime is the effect of reducing the tolerance required for qualification of a group of systems. Considering the case depicted in Fig. 2b ($m=.1$) for a training device with 30 systems, and an average designed system ($Q=.015$, $E=1$), there would be an increase in downtime of over 56% when the tolerance is tightened by 50% ($E=.5$). Downtime would increase by 83% for a very well designed system ($Q=.005$). It can be generalized that relative to an existing tolerance ($E=1$), with no change in design or operation, that increasing or decreasing the tolerance will have as a minimum, a proportional impact on downtime.

From the aspect of the production rate of pilots, there is an interesting phenomena that occurs as training time per pilot is reduced. Although this effect is not usually considered since training time is often established as a fixed period, there may be a practical limit to which pilot training time can be reduced. To demonstrate this effect, consider the following set of circumstances based on observations already made:

- o Time t_1 , to produce acceptable or unacceptable pilots, is:

$$t_1 = Nt + mnNQ/E, \text{ hrs} \quad (5a)$$

- o Number N_1 , of acceptable pilots trained, is:

$$N_1 = N - (1 - m)nNQ/E, \text{ pilots} \quad (5b)$$

- o Average training time T_a , can be calculated from:

$$T_a = \frac{t_1}{N_1} = \frac{Et + mnQT}{E - (1 - m)nQ}, \text{ hrs/pilot} \quad (5c)$$

Plotting the results of equation (5c) in Fig. 3 for average training time T_a , as a function of training stoppage proportion m ; indicates that m should be reduced whenever possible to achieve a reduction in training time. The case shown indicates a large variation, however the magnitude in reduction depends greatly on the relative magnitudes of the training time t , and repair time T . For other cases the impact of m can be larger or smaller.

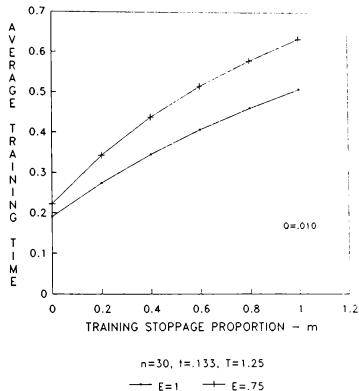


FIG. 3 - STOPPAGE IMPACT

Ideally, if the training device were to operate flawlessly, day in and day out with no downtime, the per system training rate (T_a , hrs/pilot) would exactly equal the simulator operating rate. Since training device operation is a direct result of design, engineering and maintenance, there will be significant variations between training facilities. Considering the training rate and simulator operating rate, with quality as a parameter, Fig. 4 displays typical system quality effects. Under the circumstances, it can be seen that reducing system quality results in a reduction in training rate thru-put. In fact it can be seen that for extreme cases of poor quality, there is a system operating rate ("knee" of the curve), beyond which attempting to increase operating rates results in little or no improvement in the training rate. This is the result of the proportion of downtime increasing, resulting in smaller increases in training rates. Although not plotted, tightening tolerances will show a similar reduction in training rate.

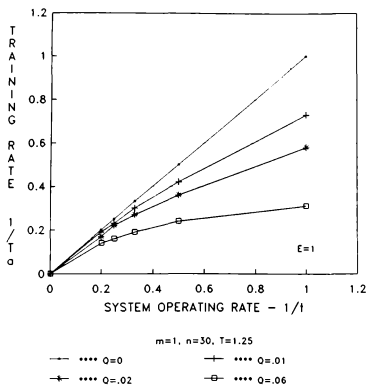


FIG. 4 - SYSTEM QUALITY EFFECT

Effect of Quality on Cost

The cost C, of each fully qualified pilot produced can be considered to be comprised of three primary components:

$$C = C_a + C_u + C_q \quad (6a)$$

C_a is the cost associated with producing only acceptable pilots, and is:

$$C_a = R(Ta) \quad (6b)$$

where:

R = Cost per unit time to operate the simulator to produce acceptably trained pilots. Includes, instructors, overheads, operating costs and depreciation, \$/hr.

C_u is the cost of correcting deficiencies associated with retraining pilots, and is:

$$C_u = UHW(Ta) \quad (6c)$$

where:

U = The number of pilots with deficiencies produced per unit time, pilots/hr.
And is calculated from:

$$U = \frac{(1 - m)nQ}{Et + mnQT} \quad (6d)$$

H = Time required to retrain deficient pilots, hrs/pilot.

W = Cost per unit time to retrain deficient pilots, \$/hr.

C_q is the cost associated with adding quality to the simulated systems via design, operation and maintenance, and is:

$$C_q = \frac{nB}{Q} \quad (6e)$$

where:

B = Quality Index, an average measure of providing quality to a simulator system, \$/system.

Substituting Eqs. (5c), (6b,c,d,e) into Eqs. (6a), and simplifying yields the total training cost of each acceptable pilot per system:

$$C = \frac{R(Et + mnQT) + (1 - m)nHWQ}{E - (1 - m)nQ} + \frac{nB}{Q} \quad (7)$$

Equation (7) shows that the per system training cost of a pilot can be broken down into:

1. A cost that decreases as quality increases.
2. A cost that will increase as quality increases.

The results of Equation (7) is depicted in Fig (5) for the typical conditions indicated. It can be seen that there exists a minimum training cost per system which varies depending on parameter values. Moving to the left of optimum results in additional cost due to improvements in quality that do not improve production rate. Moving to the right of optimum results in added cost due to additional downtime as the result of poor quality. As expected, increasing m or decreasing E results in an increase in cost. For a typical system quality $Q=.015$ and $m=.1$ there is an increase of approximately 32% in per system cost for a tolerance which is 25% tighter.

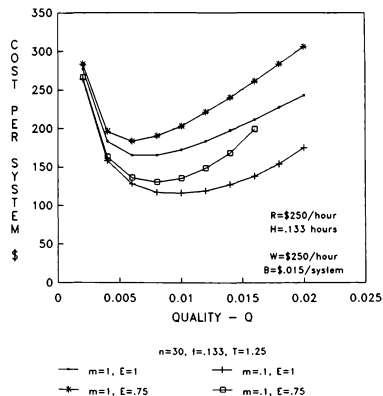


FIG. 5 - TRAINING COST PER SYSTEM

As an example of the magnitude of cost impact, consider a simulator operating 350 days per year, having thirty systems and utilizing four 4-hour training periods per day. Tightening tolerances by 25% will relate to an increase in operating cost of \$1.8 million dollars per year. Note: the analysis assumes that the tightening of tolerances occurs at the inception of the design.

To summarize overall cost; the per pilot training cost is plotted in Fig. 6 for the established conditions and a typical training period of four hours. Essentially, this is the cost to own and operate a simulator with a quality level equal to the average of the composite systems. Also plotted is the cost associated with providing a certain level of quality along with the hourly billing rate that must be charged to cover expenses.

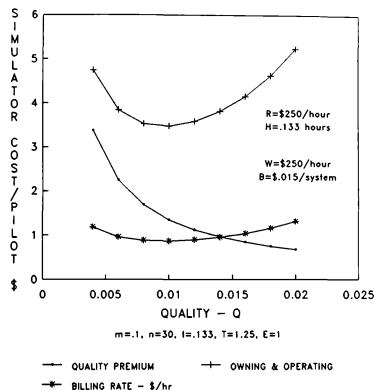


FIG. 6 - PILOT TRAINING COST

From the above analysis, it can be seen that major changes in production and cost can be effected by controlling quality and downtime. The concept of building fault tolerant systems is common in the aircraft industry but in the simulation industry is presently only applied to modeling cases where required for aircraft simulation. Fault tolerance suggests redundant systems, and as such, is prohibitively expensive in the simulation industry. Additionally, redundant systems have not been necessary to achieve the desired level of system availability. Providing a high degree of reliability in simulated systems has up to this point in time negated the need for redundancy.

Validation and Verification

The key to improved training is a balanced application of realistic physical operation coupled with appropriate

procedures. Simulator manufacturer's are primarily involved with the operational effects and only secondarily involved with procedures. The degree of fidelity required for good training has its origin in the basic set of rules used to design and build the simulation equipment. Requiring simulator manufacturer's to build equipment to tolerances beyond which a pilot cannot perceive the effect, or, tighter than the accuracy of measured data is of little additional benefit in training. Resources might better be spent on, for example: additional training, diagnostics, qualification, improved faster turn around time and applied to certain flight parameters should be a reflection of what the pilot can sense or measure. Often the case is made to tighten tolerances in order to produce better equipment. Depending on the conditions however, there may be no correlation between closer validation testing and subjective acceptance. On the other side of the coin, tolerances which are wide open can result in unacceptable or even negative training, again, being of poor value. It is important that only a judicious application of tolerances can result in the best set of circumstances for a particular situation.

The previous cost analysis considers the effects of tighter tolerances associated with validation and verification testing. It is seen that tolerance changes, complexity, design and operation quality, and, fault tolerance can have a significant effect on the production rate of acceptable pilots, and, on the associated per pilot cost. The analysis considers only the hardware operational effects of producing pilots and does not consider the ultimate quality of the produced pilot. Ultimate pilot quality has both physical and subjective components. Subjective elements must necessarily be evaluated by individual training programs.

Training Interdependency

From the volumes of information available, it is evident that the many interdependent variables involved in simulation have been studied by many hundreds of researchers and engineers. Even with this, no uniformly definitive and complete treatment of interdependent variables exists. Any discussion of training is intimately coupled with the composition of all other perceptive input systems. Definition of the human factors involved in the total system operation is critical in providing a basis from which to make decisions concerning tolerances associated with validation and verification testing. Levels of perception for the multiple operating systems need to be defined which then will provide the basis of the tolerance. Without such important information, tolerances applied during validation testing may or may not result in acceptable operation.

It is suggested that a more appropriate approach to take would entail developing a

priority listing of applicable aircraft functional systems according to training value. Next, determine the perceptive threshold levels needed for sufficient and effective training. Then, develop the appropriate tolerances needed to match the established threshold levels. Such a procedure will result in a training device with improved fidelity, and which matches more closely the objectives of hardware training without unnecessarily increasing design, build and operating costs.

Conclusions

Validation and verification testing is an essential and necessary aspect of establishing and maintaining quality training devices. The setting of tolerances associated with required validation testing is of critical importance in establishing both the quality level of the device and its owning and operating cost. It has been determined that:

1. Manufacturing tolerances for individual simulated systems must be more stringent than established regulations in order to not have negative impact on the composite simulator availability.
2. The lack of good solid mechanical and electrical design cannot be completely compensated for by imaginative software routines.
3. Accurate and high resolution data is essential for validation purposes.
4. The proportion of downtime associated with a manufactured training system is directly related to its design, manufacture and operating procedures. Implicit, is the use of high quality hardware components.
5. The production of trained pilots can characteristically be increased by:
 - o Reducing the number of operating systems per training device, i.e.; use more, lower level devices where appropriate (potential use of FTD's).
 - o Allowing training to continue on out-of-tolerance systems whenever possible and appropriate.
 - o Reduce training time by proficiency testing in lieu of fixed exposure time.
 - o Reduce the time required to correct operational problems.
 - o Open up tolerances where appropriate.
 - o Improving Systems quality.
6. The cost of pilot training can be optimized when considering the cost associated with quality and downtime.
7. Validation tolerances need to be developed from perceptive cue information and prioritized according to training value.

Acknowledgment

The author wishes to thank the Link Miles Corporation for the support and encouragement in preparing this manuscript, and, especially co-workers who reviewed, commented and provided valuable advice.

References

1. E.A. Stark, The Simulator as a Training Device. Presented at the AIAA Flight Simulation Update, State University of New York, Binghamton, New York, January 1988.
2. P.A. Lockner, and P.D. Hancock, Redundancy in Fault Tolerant Systems. Journal American Society of Mechanical Engineers, May 1990.
3. G. Boothroyd, C. Poli, L. Murch, Automatic Assembly, Marcel Dekker, Inc., New York, New York, 1982.
4. IATA, Flight Simulator Design and Performance Data Requirements, Draft No. 2 of the Third Edition, 1986.
5. U.S. Department of Transportation, Advisory Circular 120-40A, Federal Aviation Administration, July 1986.
6. U.S. Department of Transportation, Draft Advisory Circular 120-40B, Federal Aviation Administration, December 1988.
7. U.S. Department of Transportation, Advisory Circular 120-45A, Federal Aviation Administration, December 1989.

ASCENT/ABORT TRAINING IN THE SHUTTLE MISSION SIMULATOR

John T. Sims*
NASA/Johnson Space Center
Houston, Texas

Michael R. Sterling**
Rockwell Space Operations Company
Houston, Texas

Abstract

Space shuttle ascents and ascent aborts present a unique training problem. Shuttle abort profiles include scenarios that range from flights to orbit to landings at various international landing fields. Orbiter flight crews are trained in ascent and abort flight techniques at the Johnson Space Center in Houston, Texas. The Shuttle Mission Simulator is the primary tool used in this training. It employs a motion system, visual systems and a high-fidelity cockpit to enhance training accuracy. A team concept is used to train flight crews and flight controllers for each mission to ensure a thorough understanding in both orbiter systems and mission specific flight profiles.

Introduction

There are many unique aspects to the Space Shuttle program. Training for this program is no exception. The area of shuttle ascent and ascent abort training presents a most challenging simulation effort. By far the most dynamic phase of any Space Shuttle flight is the ascent

phase. It is also the most unusual and therefore the most difficult to simulate. When one considers that shuttle ascent aborts transition into shuttle entries and landings the simulation task is made even more complex.

Nominal Shuttle Ascents

A nominal Space Shuttle launch sequence begins with the ignition of the three main engines at approximately T-6 seconds. At T-0 seconds, after the shuttle main engines have reached their required power level, the solid rocket boosters (SRBs) ignite and the orbiter lifts off. Immediately after tower clear the shuttle stack rolls to a heads-down attitude and assumes its desired launch trajectory. The SRB's propellant supply is depleted after about 2 minutes into the flight at which point the SRB's separate. The shuttle then completes the ascent using only the thrust of its main engines. About eight and a half minutes after lift-off, the main engines shutdown. This is known as Main Engine Cut-Off (MECO). The shuttle with its external tank are then in a trajectory which will eventually return them to the atmosphere. The shuttle separates from its external tank which reenters the atmosphere.

* SMS Team Lead, Member AIAA

** SCA Propulsion/GNC Monitor,
Member AIAA

Shuttle Ascent Aborts

aborted is the loss of a system which is critical to the safe completion of the mission. There are five types of shuttle abort modes. They are: Abort-to-Orbit, Abort Once Around, Transatlantic Abort Landing, Return to Launch Site, and Contingency Aborts. These are illustrated in Figure 1.

An Abort-to-Orbit (ATO), the only abort ever actually performed in the shuttle program, results from a lack of performance. Most shuttle flights have the capability to perform an ATO if they lose one main engine after about five minutes into the ascent. This abort results in the shuttle being placed in a 105 nautical mile orbit, which is usually lower than its desired orbit. The flight crew and the Mission Control team time will then assess the impact of the lower orbit and replan the remainder of the mission. This is the least severe abort profile and

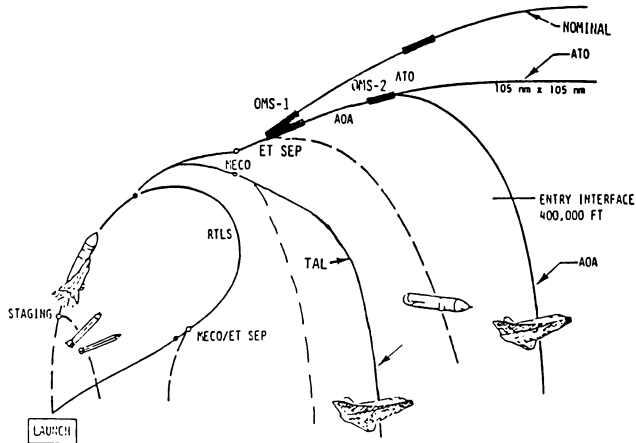


Figure 1 - Shuttle Ascent/Abort Profiles

therefore the one most desired if a nominal ascent cannot be achieved.

A more severe abort is the Abort Once Around (AOA). This abort is the result of a greater performance loss such that an ATO cannot be achieved. An AOA may also be performed for a critical orbiter systems failure, such as loss of cabin pressure integrity or loss of orbiter avionics cooling. Instead of achieving orbit the shuttle makes one near complete trip around the Earth and lands at the normal end of mission landing site, usually Edwards Air Force Base in California. This flight takes about ninety minutes to complete.

Should the performance loss occur earlier in the ascent or be even more severe, or should the systems failure be even more critical, a Transatlantic Landing (TAL) abort must be performed. Most shuttle flights have the capability to

perform a TAL abort if they lose one main engine after about three minutes into the ascent. This abort lobs the orbiter over the Atlantic Ocean for a targeted landing in any one of a number of designated landing sites in Europe or Africa. A TAL abort takes about 45 minutes to complete.

If the severity of the failures on the orbiter warrant the immediate return of the vehicle, or if the performance loss occurs very early in the ascent a Return To Launch Site (RTL) abort will be performed. This abort results in the orbiter returning to a landing site only a few miles from its original departure point. This is a very stressful abort profile which is illustrated in figure 2. The orbiter must fly out a good distance to expend fuel. Then it pitches around and uses its main engine thrust to stop and reverse its outbound velocity. The shuttle

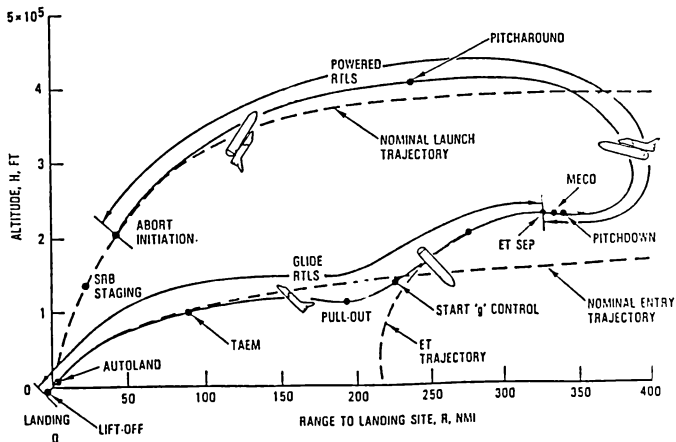


Figure 2 - RTL Abort Profile

then flies back towards the launch site until its main engine fuel is exhausted. It then separates from its external fuel tank and glides to a normal shuttle landing. Due to the severity of the flight profile and the expected stresses on the orbiter vehicle, this is the least desirable of the intact aborts.

The least desirable of all of the abort modes is the Contingency Abort or non-intact abort. A Contingency Abort would only be performed for a drastic loss of performance, such as the loss of two or all three main engines. This performance loss is so critical that no other abort can be performed. A Contingency Abort assumes that the orbiter cannot make any landing site and therefore will be lost. The concentration is then on the safe retrieval (through search and rescue operations) of the crew. Before the flight crew can bailout, they must perform a unique series of difficult procedures based on at what point they were in the ascent when the main engine failures occurred. These procedures involve manually flying the orbiter through a series of maneuvers to safely separate from its external tank and glide to bailout altitude. These maneuvers occur in flight regimes not normally seen by most flying vehicles. These flight regimes, some of which include 58 degree angles of attack, are very stressful on the shuttle orbiter.

Training for these aborts presents a unique problem. All of these aborts start as ascents, under certain ascent simulation constraints, and end as entries, under often vastly different constraints. These

flight regimes are simulated using both theoretical and wind tunnel test data and present a multitude of unique simulation tasks.

Ascent/Abort Training Facilities

All shuttle ascent/abort training facilities are located at the Johnson Space Center in Houston, Texas. The Shuttle Mission Simulator (SMS) is the primary training tool for shuttle astronaut training. It is the only high-fidelity simulator capable of training flight crews for all phases of a shuttle mission, from lift-off minus thirty minutes through touchdown and rollout. This includes prelaunch check-out, ascent, aborts, on-orbit operations, entry and landing. The SMS training complex has been fully operational for shuttle training since 1978. It consists of two training bases, a Motion Base and a Fixed Base. The motion base is used primarily for ascent and entry training, the fixed base is used primarily for orbit training, though either base can support any shuttle flight phase. A third base called the Guidance and Navigation Simulator (GNS) is used primarily for simulator software development. It is scheduled to be upgraded to a training base in 1993.

Each SMS base is supported by a dedicated Sperry 1100/92 host computer and Concurrent 3280 base intelligent controller (Base IC). The host computer contains the math models of the various shuttle systems, shuttle flight dynamics models, and atmospheric models. The Base IC supports the host computer by acting as an input/output processor for the various

simulator interfaces. This function includes providing for instructor and simulator operator inputs and displays, flight crew inputs and displays, as well as providing the interface between the models running in the host and the Network Simulation System (NSS). The NSS simulates the shuttle tracking and communications network for integrated training with the Mission Control Center. The Base IC also serves as the interface between the host computer and the Simulation Interface Device (SID). The SID contains a dedicated set of five shuttle General Purpose Computers (GPCs). These GPCs are nearly identical to those used in the real vehicle and they operate with the real flight software to be used for the particular mission being simulated. This is very important as the SMS is the only training facility that uses real flight software.

Each base is equipped with a visual system. This system provides computer generated images of the shuttle launch tower and Kennedy Space Center, various shuttle landing sites around the world including Edwards Air Force Base, Ben Guerir, Morocco, Moron, Spain, and Hawaii. For orbit simulations the visual system can display payload bay scenes as well as accurate star fields.

The fixed base simulator consists of high-fidelity mockup of the shuttle orbiter flight deck and middeck in their flight configuration. There are two parts to the motion base simulator. The first is the orbiter flight deck. This consists of a high-fidelity simulation of the shuttle orbiter forward flight deck displays and controls and

four flight-type seats for the shuttle crewman. The motion base flight deck is mounted on a six degree of freedom motion platform. The motion base also has a simulation of the orbiter aft flight deck and middeck located next to the motion platform. The aft flight deck and middeck are accessible to the forward flight deck via a bridge whenever the motion base is used for orbit simulations.

The motion system is similar to motion systems used in aircraft simulators throughout the country. One major capability has been added however. This is the capability to pitch the simulator platform up 90 degrees to simulate the shuttle launch position. This capability is called "extended pitch". This is illustrated in figure 3. Extended pitch is used throughout the ascent simulation. Prelaunch the motion base is in full extended pitch to simulate the orbiter's position on the pad. Extended pitch is maintained until SRB separation when pitch is reduced suddenly to simulate the reduction of G forces that occurs at this time. Pitch is gradually increased to simulate the increase in G forces that occurs as the shuttle continues its ascent. Other shuttle motions are simulated in the same manner using "motion cues". The motion system generates a "motion cue" by an initial movement in the appropriate direction. Once the initial motion is complete the motion system returns to its neutral position to be ready for the next cue. The illusion of continuous movement is maintained by the visual system which maintains this through the apparent motion of the visual scene.

Training Facility Advantages and Disadvantages

There are several advantages to training in the SMS. The first of these is the motion capability. The motion system aids in familiarizing the crew with the environment they will be experiencing during the actual ascent. Another advantage is the ability to train the entire shuttle mission in one facility. For example, one can perform an entire AOA abort from launch through the orbit and deorbit phase to entry and landing an hour and a half later. Finally the fidelity of the SMS is high. The cockpit hardware is identical to that used in the real vehicle and the software models have a high degree of accuracy considering the complexity of the modeled systems.

The SMS is not without its disadvantages however. The visual system, while state-of-the-art when it was installed in 1978, is obsolete and has limited capabilities. A new modern visual system is sched-

uled to be installed in 1993. Another difficulty with the SMS is the amount of time required to generate simulator software. For each shuttle flight a package of simulator software, called the training load, is delivered. The training load contains the flight software for the mission, the payload models and the standard orbiter systems models. Training loads currently take many months to generate and require several hundred hours of SMS time to complete for each flight. Model updates and fixes to systems model problems are delivered twice a year. Therefore, the time between when a problem is identified and when the fix is delivered to the SMS is often very great.

SMS Instructor Teams

The SMS is used for shuttle ascent/abort familiarization and systems training. During ascent/abort training sessions, the crew is subjected to four hours of repeated ascent runs. The primary purpose of this

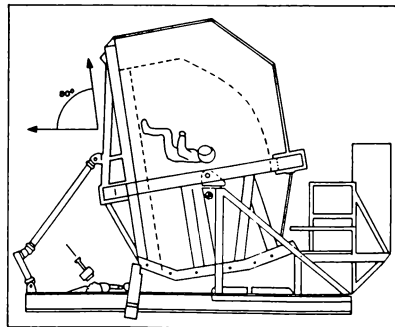
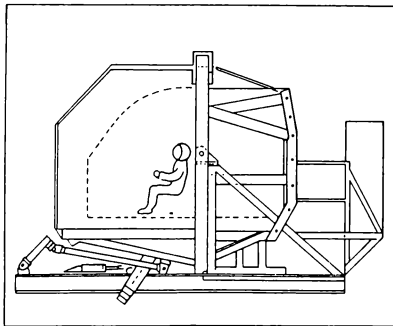


Figure 3 - Motion Base Crew Station Extended Pitch

type of training is to acquaint the flight crew with methods of dealing with mechanical systems failures and other types of problems during this very dynamic phase of flight.

Each training session on the SMS is supported by a team of instructors. The basic SMS team consists of five people, the team lead, the control/propulsion instructor, the systems instructor, the data processing/navigation instructor, and the communications/instrumentation instructor. The team lead is in charge of the SMS team. It is his or her job to coordinate the other instructors actions, ensure the training session is conducted in an orderly and timely manner, and that the training requirements for the lesson are met as defined in the Shuttle Crew Training Catalog. The control/propulsion instructor is responsible for training the shuttle's propulsion systems, such as the reaction control system and the orbital maneuvering system, and for training the shuttle's control systems, such as the digital auto pilot and the flight control system. The control/propulsion instructor is also responsible for instructing in shuttle guidance and flight techniques. The systems instructor is responsible for training all the orbiter's mechanical, electrical and environmental systems, such as the auxiliary power units, the payload bay doors, and the fuel cells. The data processing system/navigation instructor is responsible for the training of the orbiter's general purpose computers and their associated data buses as well as the shuttle's navigation equipment. The communications/instrumentation instruc-

tor is responsible for training in the orbiter's data and voice communications systems and the orbiter's data collection instrumentation system. One team of instructors is assigned to each shuttle mission. The assigned team supports every SMS session in which their crew participates. In addition to the above instructors there are several specialty instructors who support SMS training as needed.

During a typical ascent session several different runs are made starting a few minutes before lift-off and ending on-orbit for an "uphill" run or after landing for an abort run. During each run each instructor inputs failures for his particular system based on a lesson plan, called a script, the instructors have written in preparation for the training session. The crew reacts to these systems failures and works the proper procedures in response. These failures often require an ascent abort. The instructors monitor the crew's actions in response to the failures and advise and critique as necessary. The instructors also monitor the crew's performance of nominal shuttle procedures.

Specialty Instructors

Some elements of Space Shuttle mission training are so specialized that instructors must devote a great deal of time just to stay abreast of any changes in either orbiter technology or flight procedures. In order not to burden every instructor with the responsibility of studying each unique topic in depth, specialty topics are assigned to one or more instructors. Although

these assignments are made in addition to their normal responsibilities, by spreading these special topics out over a number of instructors, it is possible to maintain an adequate level of expertise in the specialty topics without undue burden on the bulk of the instruction staff. The two most notable of these specialty topics are space shuttle landing and rollout, and Contingency Aborts.

A shuttle landing and rollout specialty instructor is required due to the unique aspects of shuttle landings. Although the Shuttle appears to land like any other aircraft, there are some distinct differences. For example, the Shuttle approaches at a 19 degree outer glideslope and transitions to touchdown attitude in a matter of seconds, allowing it to land at 195 knots. Also, the steering and braking systems on the orbiter are being improved gradually as the shuttle program matures. All flight crews must be made aware of these and other changes in landing and rollout procedures and systems.

Another aspect of Space Shuttle flight that is undergoing growth is the area of Contingency Aborts. As described earlier, Contingency Aborts require a substantial amount of training due to their many specialized procedures. Continual training on various training and engineering simulators at the Johnson Space Center brings to light new and improved methods of flying this very labor intensive piloting task. Keeping appraised of the specifics of this specialty topic require the continual attention of a member of the instructional staff.

"Pilot Pool" Training

SMS training sessions are divided into two categories. These are "pilot pool" sessions and "mission specific" sessions. A "pilot pool" session is one where the crew to be trained is made up of astronauts not assigned to an actual flight crew. Usually, the "pilot pool" crew consists of several newer astronauts and one astronaut who has flown one or more flights. "Pilot pool" training is designed to accomplish two tasks. The first of these tasks is to improve the astronaut's understanding of Space Shuttle systems operations, and expose them to shuttle crew coordination. The second task of these sessions is to maintain proficiency in shuttle operation among the senior astronauts.

"Pilot pool" sessions are structured to allow less experienced astronauts to draw on the experiences of their seniors. This is accomplished with the help of a senior astronaut acting as an instructor pilot. During "pilot pool" training sessions, the senior astronaut is expected only to supplement the training of his less experienced colleagues. The responsibility for the training accomplished in these sessions belongs to the SMS training teams.

Mission Specific Training

Once a group of astronauts is assigned to a flight they begin mission specific training. Training usually starts approximately eight to nine months before the scheduled launch date with "flight-similar" training. "Flight-similar" training occurs at the SMS

using simulator software from a previous shuttle flight that has a similar ascent/entry profile. The first part of the crew's training is referred to as stand-alone training. In stand-alone training the instructors act as mission control, advising the crew, making normal ground calls, and making abort calls. The real flight controllers are not involved in these lessons. Therefore the lessons concentrate on procedures and malfunctions that the flight crew can work with a minimum of ground information. Since this is "flight-similar" training with a training load from a different flight the emphasis of these integrated sessions is on solving systems problems rather than completing specific mission objectives.

The flight crew continues "flight-similar" training until eleven weeks before their scheduled launch date. At this time their "flight-specific" training load is delivered. This training load is tailored to match their flight. It contains the flight's ascent profile, payloads and flight software. At this point the crew begins to concentrate on unique aspects of their ascent profile and abort procedures.

Integrated Training

In addition to stand-alone training sessions in the SMS, unassigned astronauts and assigned crews, participate in integrated training sessions. In an integrated session, the SMS is linked to the Mission Control Center (MCC) by means of the Network Simulation System (NSS) which transmits data from the SMS to the MCC simulating the real shuttle vehicle's telemetry stream. An

integrated training team mans the Simulation Control Area (SCA) in addition to the training team at the SMS. The SCA is located near the mission control room. Each SMS instructor has a counterpart in the SCA and they are in constant communication during the training session. It is the SCA instructors' job to monitor the performance of the individual members of the flight control team.

There are two types of integrated ascent simulations, generic and flight specific. To allow unassigned and often inexperienced flight controllers to become accustomed to the dynamic atmosphere of space shuttle mission monitoring, generic integrated simulations were developed. These sessions place flight controllers in the MCC monitoring unassigned astronauts, or recently assigned crews, performing non-mission specific activities in the SMS. There are generic sessions for ascent, entry, and orbit activities.

When a shuttle mission is manifested, both a crew and a set of flight control teams is assigned to that flight. Many of the members of the flight control teams are working other flights at the time of assignment and must complete these previous duties before, and often while, beginning these new responsibilities. As the assigned crew enters the flight specific portion of their mission training, they and their assigned flight control teams are given flight specific integrated simulations. These sessions serve a number of purposes. They allow both the crew and the flight control team to perform nominal and off-nominal flight specific

activities and become accustomed to those activities. They allow the flight controllers to get used to the specific crew members and their responsibilities on the flight. They allow the crew to develop confidence in the flight control team and they allow the necessary repetitive practice of flight specific critical activities.

Each flight usually has about four integrated ascent training sessions. These sessions are usually six hours long and follow the same format as the stand-alone ascent training sessions. The only difference is that the real flight control team participates instead of the SMS instructors making their calls. During these sessions, every sort of ascent and abort scenario possible for that mission is covered in depth. In addition, for each flight a dedicated integrated training session is scheduled specifically for only those flight controllers involved in the monitoring of the performance of the space shuttle main engines and the ascent guidance scheme. These are called Flight Dynamics Officer/Booster Systems Engineer (FDO/BSE) sessions. The FDO/BSE training session concentrates on abort profiles and main engine failures without the complication of other systems failures.

Advantages and Limitations of the Training Methods

There are several advantages and limitations to the training techniques used in shuttle ascent/abort training. The Space Shuttle is the most complex flying vehicle ever built. Its designed flight regimes encompass everything

from 195 knot atmospheric flight to 25,000 foot per second orbital space flight. As one might expect, it is very difficult for one person to maintain an intimate knowledge of all of the systems on a vehicle of this complexity. The current system of instruction used by NASA at JSC was adopted to ensure thorough instruction in each of the systems of the orbiter. For example, if an instructor is expected to have a thorough knowledge of the shuttle communication systems but not the propulsion systems, then that instructor should be able to impart a great deal of knowledge concerning the communication systems. By using this method of "team" instruction in the SMS for crew training and in the MCC for flight controller training, it is felt that the students are presented with the maximum amount of technical information and familiarization with flight specific activities while at the same time stressing the idea of teamwork.

One of the unfortunate aspects of the team training concept is that it relies heavily on the strength, both of personality and technical knowledge, of the individual members of the training team. An instructor with an excessively strong personality will tend to over stress their particular area of expertise at the cost of other training. Conversely, an instructor who is not as aggressive will occasionally not stress their systems enough. It is therefore difficult to maintain an acceptable training balance. It is more common to see training scenarios which strive to teach something about every area of specialty in a single

ascent and result in a case that is far more complex than ever intended. This points out the need for a strong SMS Team Lead or Simulation Supervisor (the SCA equivalent of a Team Lead). Their roles are far from merely administrative. They are truly responsible to insure that each crew leaves for the launch complex fully trained for their mission.

Conclusion

Shuttle ascents and aborts present a unique simulation and training challenge. This challenge has been met using conventional simulation technology with some modifications and somewhat unique training philosophy and methods. Shuttle flight crews and flight controllers are trained using a team instructional approach to insure a thorough understanding of shuttle systems and ascent and abort profiles. Each flight crew and flight control team is well prepared for the actual flight by practicing over and over again both the nominal ascent and each type of abort possible for that shuttle mission.

POWER SPECTRAL ANALYSIS TO INVESTIGATE THE EFFECTS OF
SIMULATOR TIME DELAY ON FLIGHT CONTROL ACTIVITY

Matthew S. Middendorf
Steven L. Lusk
Logicon Technical Services, Inc.
P.O. Box 317258
Dayton, OH 45431-7258

Capt. James D. Whiteley
Armstrong Aerospace Medical Research Laboratory
Wright Patterson Air Force Base, OH 45433-6573

Abstract

In a recent experiment at the Harry G. Armstrong Aerospace Medical Research Laboratory, Human Engineering Division, test subjects were instructed to perform a sidestep landing maneuver in a flight simulator with time delays of 90ms, 200ms, and 300ms. The baseline delay condition was 90ms and additional delays were added to the visual display loop to yield the 200ms and 300ms delay conditions.

Power spectral analysis on lateral stick activity showed that power in a narrow band (0.4 to 0.5 Hz) increased as time delay increased. We have examined this increased power and determined that, as time delay increased, the man-machine system became less stable and less damped. Thus, the subjects needed to make additional control inputs to correct for overshoot and degraded stability.

Introduction

The use of real-time man-in-the-loop flight simulators for pilot training has steadily increased over the past few decades. The trend in simulator evolution has been towards higher aerodynamic fidelity and increased realism (i.e., scene complexity). These two factors have

led to increased computational requirements, and in many cases, increased simulator time delay, which if not properly controlled, can have deleterious effects on simulator training effectiveness. ¹ Time delay as used here is defined to be the delay between pilot input and pilot cuing by the visual display, excluding delay due to vehicle dynamics (i.e. phase shift). ²

Previous research in our laboratory has used performance measures such as root-mean-square (RMS) error (e.g., deviation from an assigned altitude) to evaluate the effects of increased simulator time delay. In this paper we explore the use of power spectral analysis to investigate the effects of increased delay on pilot control activity.

Methods

Subjects

Twelve male college-age volunteers were paid to participate in the experiment. All had normal or corrected 20/20 visual acuity and were right-handed. All subjects were naive to the sidestep landing maneuver, but were experienced in a heading and altitude disturbance regulation task that utilized the same hardware. ³

This paper is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

Apparatus

Simulated Aircraft. The experiment was conducted in a fixed-base simulator with a moderate-fidelity aerodynamic math model. The strategy for developing the math model was to start with an accurate and complete manufacturer's data source for a modern fighter aircraft, the F-16.⁴ The data were then interpolated and linearized about a fixed operating point, the landing approach configuration. The simulated aircraft was assumed to have fixed mass and its motion was governed by standard rigid-body force and moment equations.⁵ The aerodynamic math model was implemented on a Digital Equipment Corporation Microvax II.

The simulated aircraft was controlled by a side-mounted isometric force stick. The stick inputs were scaled with a single gradient function after a 2.2 newton breakout force. Thrust changes were accomplished using a linear, single-engine throttle. The throttle drove two first-order transfer functions that changed the throttle to fuel flow and fuel flow to thrust, with appropriate characteristics for the

simulated aircraft. A sound generator was used for throttle feedback. Three frequencies were blended together corresponding to air speed, engine RPM, and rumble.

Displays. The primary display was an out-the-window scene generated by a Silicon Graphics Iris 3130. The scene consisted of two parallel runways with visual approach slope indicator (VASI) lights located on both sides of both runways (figure 1). The scene was projected onto a flat, matte-white screen by a model IIc Aqua Star projector. The scene was 6 ft (1.84 m) high by 8 ft (2.46 m) wide with a corresponding field of view of 58 degrees vertical by 74 degrees horizontal. The secondary display was an instrumentation display generated by a Silicon Graphics Iris 3120 and presented on a high-resolution 13 inch (33 cm) color monitor. The instrumentation included an altimeter, attitude directional indicator (ADI), airspeed indicator, fuel flow indicator, compass, and a vertical velocity indicator. The time delays for the primary scene and the instrumentation display were the same.

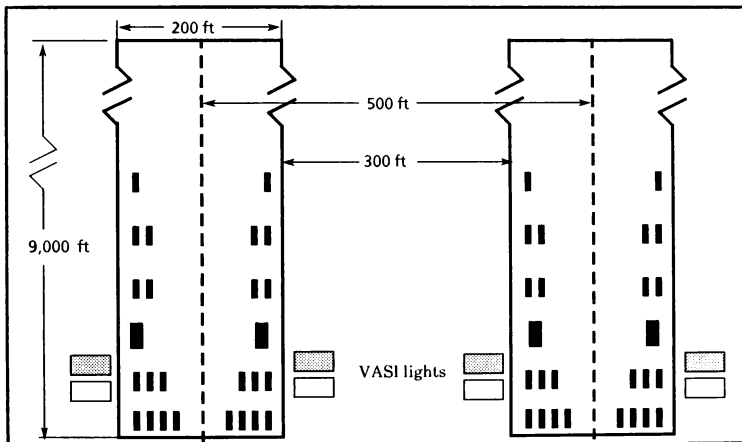


Figure 1. Top view of runway setup

Delay Verification. To measure the simulator time delay, several sinusoidal test frequencies were substituted for stick inputs. A photocell was used to measure the simulator response on the visual display. The phase shift between the input and the output was determined by a frequency response analyzer (Bafco model 916). The phase shift due to the aircraft dynamics was subtracted from the measured value at each of the test frequencies. The remaining phase shift and the input frequency were used to calculate the simulator time delay. The minimum time delay that we were able to achieve was 90 ms, the baseline condition. The 200 ms and 300 ms delay conditions were generated by adding delay in the software. For a more detailed description of the delay verification process the reader is directed to Johnson et al. (1988).² The 77 ms equivalent delay associated with the control surface servo-actuators of the simulated aircraft, were nulled so that the remaining, 90ms, baseline simulator time delay closely approximated the actual aircraft.

Task Description. The simulated aircraft started in a trimmed final approach configuration, with landing gear down, at a distance of 10,000 feet from the runway threshold. The initial airspeed was perturbed by ± 10 knots to encourage the subjects to engage in active control from the beginning of the approach. The subjects were encouraged to correct the airspeed perturbation and resulting glide slope deviation before being cued to perform the sidestep to the adjacent, parallel runway. To prevent subjects from identifying a fixed reference point on the display and anticipating the cue, 10 discrete points within a 1000 ft (307.2 m) window were selected for cue presentation. The order of the 10 discrete points was randomized and balanced. In addition, straight-in approaches (i.e., no sidestep) were randomly inserted to further ensure that subjects were not anticipating the cue.

The aircraft was initially aligned with the right runway, on glide slope and ground track. At one of the ten discrete points described above, a red X appeared on the right runway. The X cued the subjects to perform the sidestep maneuver. The subjects had a fixed distance (1500 ft) to acquire alignment with the

left runway (figure 2). Subjects learned to follow a fixed approach by using the VASI lights and a feedback display. After each trial, the feedback display was presented on the instrumentation monitor. It depicted the desired approach (ground track and glide slope) and the subjects' actual approach (figure 3). In addition, numerical feedback was presented on the primary display. This included mean, standard deviation, and RMS scores for glide slope error and ground track error.

Procedures

Experimental Design. The subjects performed five familiarization trials before data collection began. The data collection was separated into two phases. The first phase was a quasi-transfer-of-training design and the second was a repeated measures, multiple 3 X 3 latin square, within-subject design. In the first

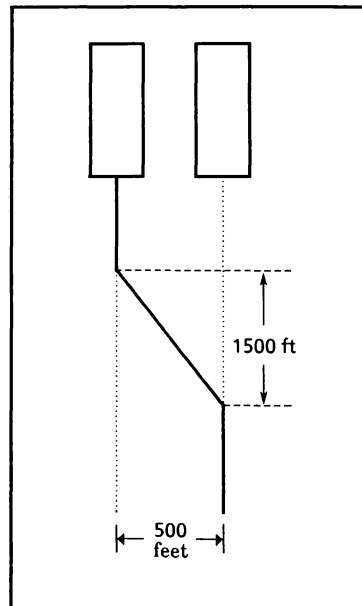


Figure 2. Re-alignment Distance

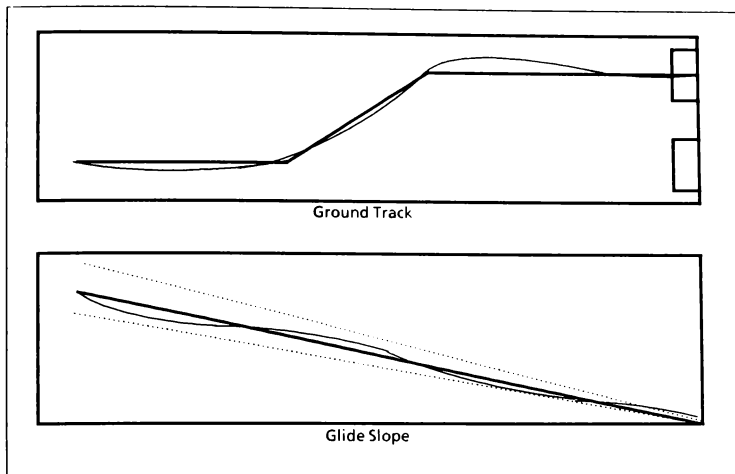


Figure 3. Typical feedback display following each trial. Subjects were provided feedback which showed their actual performance vs. the desired performance for glide slope and ground track. The thick solid lines represent desired performance; the thin dotted lines on the glide slope display represent the tolerable error that will not cause the VASI lights to indicate a change in glide slope.

phase, subjects were divided into three groups and each group completed 40 training trials in one of the three delay conditions (90, 200, 300ms). Then all three groups transferred to the minimum delay condition (i.e. the condition that most closely resembles the actual aircraft). In the second phase of the experiment, subjects completed 40 trials in each of the three delay conditions. The order of the delay conditions was determined by the latin square.

The quasi-transfer-of-training design is consistent with previous research designs used in our laboratory.⁶ However, due to the complex nature of the sidestep maneuver, the subjects did not reach asymptotic performance after just 40 trials. Therefore, the power spectral analysis reported in this paper uses only the 120 trials (40 per delay condition) from the latin square portion (phase II) of the

experimental design. The straight-in approaches were excluded from the analysis.

Fourier Analysis. The fourier analysis was accomplished using the SAS/ETS software package licensed by the SAS Institute, Incorporated. The output from this software was independently verified using an in-house software package. A power spectrum for lateral stick activity was generated for each individual trial in the latin square portion of the data collection, excluding straight-in approaches. On the average, 3,235 data points were collected per trial. The number of points per trial varied due to changes in airspeed and touchdown point. As a result, a 2,048 point window was selected so that it always captured the sidestep portion of the maneuver.

Results

Three average power spectra were created for each subject, one per delay condition. These individual power spectra were then averaged across subjects by delay condition. Three interesting results surfaced. First, there is a peak in the power spectrum at approximately 0.08 Hz (figure 4). Further investigation, using time history data, revealed that it takes approximately 12 seconds to complete the sidestep maneuver. This may be seen in figure 5 which shows lateral stick activity and aircraft roll angle for a representative trial (e.g. one that closely matches the average). The 0.08 Hz component, due to the maneuver itself, is not obvious in the lateral stick time history. However, when looking at aircraft roll angle the 0.08 Hz component becomes more apparent.

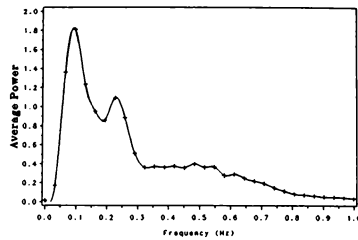


Figure 4. Average power at 90 ms

The second result is another peak at approximately 0.25 Hz (figure 4). Once again, this peak appears to be a direct result of the maneuver itself. Performing the sidestep

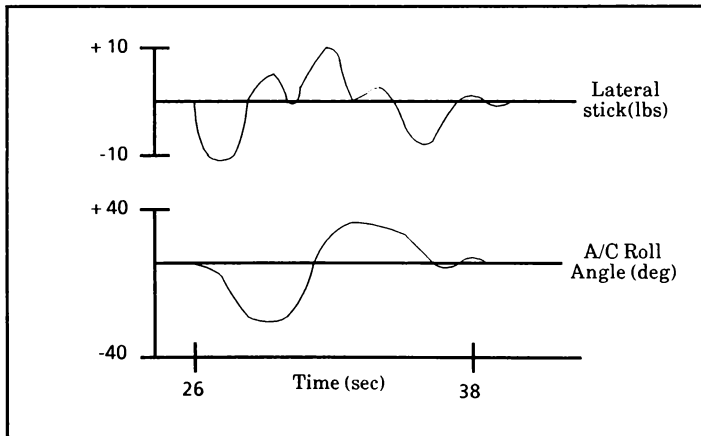


Figure 5. Typical trial

requires three inputs: 1) roll left to change heading to intercept the localizer of the left runway, 2) roll right to bring the heading parallel to the localizer, and 3) roll left to bring the aircraft back to level (figure 5).

The third finding is the most interesting because it illuminates the deleterious effects of simulator time delay. Power in the 0.4 Hz to 0.5 Hz range significantly increased ($F(2,22) = 26.97, p < 0.0001$) as time delay increased (figure 6). In addition, power at the 0.25 Hz peak tended to increase ($p > .05$) with time delay, suggesting that the subjects may be overshooting the desired bank angle for the three phases of the maneuver. The overshoot due to increased time delay is consistent with modern control theory⁷ which states that as time delay in a closed-loop system increases, the damping of the system decreases. The reduced damping results in the closed-loop man-machine system becoming less stable at the higher delay conditions.

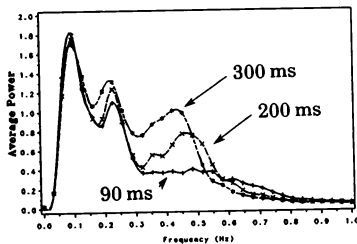


Figure 6. Average power at 90, 200 and 300 ms

The decreased stability and damping requires the subjects to work harder to perform the task. It has been shown earlier that the subjects need to put energy into the system at 0.08 and 0.25 Hz to perform the task. Therefore, the increased power in the 0.4 to 0.5 Hz region represents additional control effort by the subjects and can be regarded as an artifact of increased time delay.

Discussion

While there was no zero delay case for comparison, a simulator time delay of 90 ms does not appear to degrade the control behavior of the subjects in the sidestep landing maneuver. That is, power is at the appropriate frequencies for smooth execution of the maneuver. However, when delay is increased to 200 ms, control activity increases at frequencies not required to perform the task. In addition, power at the frequencies necessary for task execution also increases, suggesting that subjects are overshooting their desired bank angle. These two artifacts of increased time delay are exacerbated at the 300 ms delay condition. These findings are corroborated by examination of time history data. In conclusion, when studying the effects of simulator time delay, frequency domain analysis may be used, in conjunction with other types of measures to explore the manner in which time delay effects control behavior, performance, and workload.

Acknowledgments

The authors would like to thank Dr. Grant McMillan for his guidance and support. We would also like to thank Jeff Cress, Sridhar Adapalli, Jeff Wood, and Robert Jones for their assistance.

References

1. Riccio, G.E., Cress, J.D., Johnson, W.V. (1987). The Effects of Simulator Time Delay on the Acquisition of Flight Control Skills: Control of Heading and Altitude. Proceedings of the 31st Meeting of the Human Factors Society. New York, NY: Human Factors Society

2. Johnson, W.V. & Middendorf, M.S. (1988). Simulator Transport Delay Measurement using Steady State Techniques. AIAA paper 88-4619-CP, Atlanta, GA.
3. Lusk, S.L., Martin, C.D., Whiteley, J.D. & Johnson, W.V. (1990). Time Delay Compensation, Using Peripheral Visual Cues, In an Aircraft Simulator. Manuscript submitted for publication.
4. Heffley, R.K. (1988). Flight Simulation Laboratory Math Model Development and Verification. CR-RHE-SRL-87-2
5. McRuer, D., Ashkenas, I., & Graham, D. (1974). Aircraft Dynamics and Automatic Control. Princeton University Press, Princeton, NJ.
6. McMillan, G.R., Martin, E.A., Flach, J.M., & Riccio, G.E. (1985). Advanced dynamic seats: An alternate to platform motion? Proceedings of the 7th Interservice/Industry Training Equipment Conference. (pp 153-163). Arlington, VA: The American Defense Preparedness Association.
7. Kou, B.C., (1987). Automatic Control Systems. Prentice-Hall, Inc., Englewood Cliffs, N.J.

DYNAMIC SEAT CUING WITH WIDE VERSUS NARROW FIELD-OF-VIEW VISUAL DISPLAYS

Grant R. McMillan*

Armstrong Aerospace Medical Research Laboratory
Wright-Patterson Air Force Base, OH 45433

Jeffrey D. Cress

Logicon Technical Services Inc.
Dayton, OH 45433

Matthew S. Middendorf

Logicon Technical Services Inc.
Dayton, OH 45433

Abstract

Numerous studies performed in our laboratory have demonstrated that dynamic seats (g-seats) can provide highly effective roll and pitch cues for turbulence-regulation flight-control tasks. All of this research was performed using narrow field-of-view (FOV) visual displays. Although there is little empirical support for the assertion, many in the simulation community argue that wide FOV displays minimize the need for motion cuing. The belief is that peripheral visual cues can substitute for the vestibular and tactile/kinesthetic cues provided by a motion system. In the experiments reported here, we evaluated the combined effects of dynamic seat cuing and display FOV on the performance of a heading and altitude control task. We found that dynamic seat cuing significantly improved subject performance with narrow (21° by 28°) and wide (60° by 83°) FOV displays. However, in both studies performance was better with the limited FOV displays.

Introduction

Dynamic seat research at the Armstrong Aerospace Medical Research Laboratory, Wright-Patterson Air Force Base, OH has evaluated the

capability of a hydraulically-actuated seat to provide onset motion cues. This is the same type of information platform motion systems are designed to provide. These studies have clearly demonstrated that the choice of drive algorithm is critical.^{1,2} Fortunately, the same algorithm is effective for all angular degrees of freedom.^{3,4} This algorithm moves the seat elements in proportion to a weighted combination of aircraft angular position and angular velocity. Using this drive law, we have observed significant improvements in roll, pitch, heading, and altitude control for turbulence-regulation tasks.⁴

All of the above research was conducted with narrow field-of-view (FOV) displays -- 22° vertical (V) by 30° horizontal (H) or smaller. As the next step in this program of research, we intend to evaluate the contribution of the dynamic seat to more complex/realistic tasks such as air refueling or terrain following. Since pilot comments suggest that a wide FOV is necessary these tasks, we plan to use a larger display in these studies.

Previous research has identified several cases where increasing the display FOV enhances pilot performance. Gray⁵ demonstrated that a wide FOV improved pilot control of the A-10 aircraft in simulated manual reversion mode, a degraded control condition. Larger displays have been shown to improve carrier landings,⁶ helicopter shipboard landings,⁷ and basic flight maneuvers.^{8,9} Irish et al.⁸ pointed out the task

* Member AIAA

dependence of the FOV effect. While takeoff, ground controlled approach, overhead pattern, and aileron roll performance was usually better with the larger display (150° V by 300° H), performance of most aspects of a slow flight task was actually better with the narrow FOV condition (36° V by 48° H). The authors noted that a wide FOV seemed most important for tasks requiring precise roll-axis control. Irish and Buckland⁹ found similar results in a study which used a more sensitive experimental design. Display FOV affected performance on four of the five maneuvers tested. For these maneuvers, roll control was typically better with the large FOV, while pitch control was often better with the small display.

Many in the simulation community argue that wide FOV displays reduce the need for motion cuing. The belief is that peripheral visual cues can substitute for the vestibular and tactile/kinesthetic cues produced by a motion system. This belief represents a questionable extrapolation of research which has shown that vestibular information and peripheral visual information are important for, and have similar effects on, the perception and control of self motion.^{10, 11} For example, stimulation of either sensory system can produce strong sensations of self motion.¹¹ In addition, adding peripheral displays or a motion system to a narrow FOV display decreases the time required by pilots to judge their roll rate and to predict the effect of step inputs.¹⁰ Finally, peripheral displays and motion tend to have qualitatively similar effects on pilot flight-control behavior.^{10, 12} However, one cannot conclude from these findings that peripheral displays eliminate the need for, or even reduce the effectiveness of a motion system.

Unfortunately, the FOV studies reviewed above did not answer this question. The Irish et al. studies addressed the interaction of motion and FOV, but platform motion typically degraded pilot performance in their simulations. Thus, the results are not particularly germane.

The most relevant work is that of Hosman and van der Vaart,¹⁰ who evaluated several combinations of a central visual display, a peripheral visual display, and a platform motion system during the performance of two roll-axis tracking tasks. The central display (9° V by 7° H

FOV) consisted of an aircraft symbol and a moving horizon line. The peripheral displays consisted of moving checkerboard patterns presented on large TV monitors located on each side of the subject. These displays provided roll-rate information, only. The motion system provided high-fidelity roll-axis cues. The authors found that performance of both tasks was improved significantly when the peripheral displays were added to the central display. When motion was added to the central display alone (narrow FOV condition), tracking performance on the disturbance regulation task improved by approximately 60%. When motion was added to the combination of central and peripheral displays (wide FOV), performance improved by approximately 56%. A similar trend was observed for the target-following task, although the motion benefit was not as large. Thus Hosman and van der Vaart demonstrated that motion cuing enhanced performance under both narrow and wide FOV conditions, for these two tasks.

The experiments reported here used a more complex disturbance-regulation task than that of Hosman and van der Vaart. Subjects were required to maintain an assigned heading and altitude in the presence of pseudo-random roll and pitch disturbances. In this task, roll control is critical for heading maintenance. Based upon the research reviewed above, we expected the following results in the present experiments: (1) Dynamic seat cuing would significantly improve roll, pitch, heading, and altitude control under both the narrow and wide FOV conditions, and (2) increasing the display FOV would significantly improve roll and heading control. In Experiment 1, the narrow FOV condition was presented on a high resolution color monitor, the same type used in previous dynamic seat studies. The wide FOV condition was produced with a color CRT projector and flat, matte-white screen planned for use in future research. In Experiment 2, both FOV conditions were generated with the CRT projection system to ensure identical brightness, contrast, and resolution for the two displays.

Methods

Subjects. Six subjects with normal or corrected-to-normal vision participated in Experiment 1.

Graphics IRIS 2400 Turbo) and the out-the-window scene was depicted in one of the following manners: For the narrow FOV setup used in Experiment 1, the scene was depicted on a high-resolution color monitor. The image on the monitor measured 0.27 meters high by 0.36 meters wide and was located approximately 0.71 meters from the subject, resulting in a 21° V by 28° H field of view.

The wide FOV setup used an Electrohome projection system. The image was projected onto a flat, matte-white Da-Lite screen and measured approximately 1.83 meters high by 2.82 meters wide and was viewed at a distance of 1.60 meters which yielded a 59.5° V by 82.8° H field of view. The wide FOV condition was the same for both studies.

The narrow FOV condition in Experiment 2 also used the Electrohome projection system. The image measured approximately 0.52 meters high by 0.80 meters wide and was viewed at a distance of 1.60 meters which yielded a 18° V by 28° H field of view.

Dynamic Seat. The Advanced Low-Cost Guiding System (ALCOGS) is a device that can display both onset and sustained motion information (Figure 3).¹⁴ The seat pan is supported by three hydraulic actuators, one at each of the rear corners and one in the center at the front of the seat. The rear actuators were used to roll the seat about the longitudinal axis, while the front seat pan actuator pitched the seat about its rear edge. There are also three actuators for

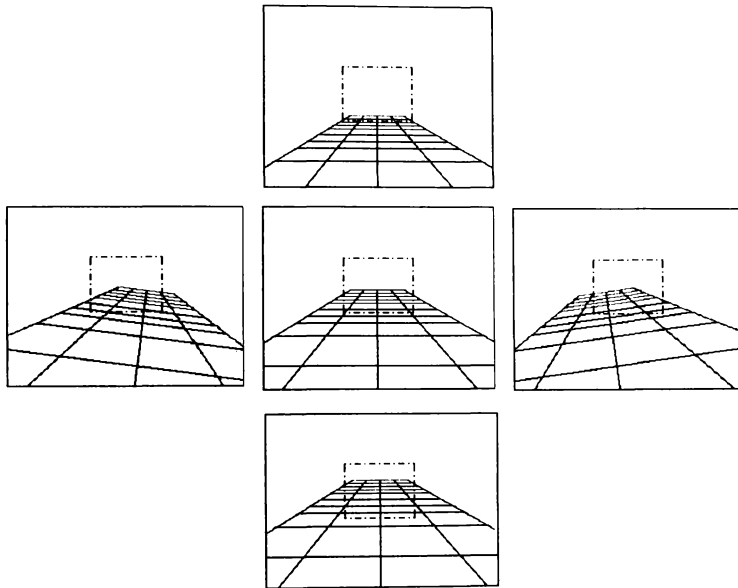


Figure 2. Visual Scene. The small box in the center indicates the display area of the narrow FOV condition.

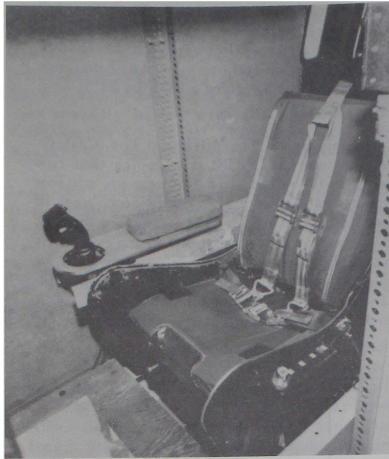


Figure 3. Dynamic Seat

the backrest, one located at the top center and the other two located at the bottom corners. The top backrest actuator (the only one used for this study) pitched the backrest about its bottom edge. The seat pan and backrest pitched as if they were a single unit.

Roll and pitch seat motion were used in this study. The drive algorithm for the seat was based on the zero and first derivatives of the simulated aircraft's (A/C) roll and pitch.¹⁴ The actual equations were as follows:

$$\text{seat roll angle} = K1 \times [A/C \text{ roll angle} + K2 \times A/C \text{ roll rate}]$$

$$\text{seat pitch angle} = K1 \times [A/C \text{ pitch angle} + K2 \times A/C \text{ pitch rate}]$$

$$\text{where: } K1 = 0.335 \quad \text{and} \quad K2 = 0.167$$

Task Description. Subjects were told to fly parallel to the longitudinal grid lines (i.e. maintain heading) and to maintain an altitude of 30.5 meters in the presence of the atmospheric

disturbances. The mean, standard deviation and root-mean-square values for the heading and altitude errors were presented to the subjects at the end of every trial.

Procedures

Experimental Design. A completely within-subjects design was used for both studies. All subjects performed the heading and altitude disturbance-regulation task with and without seat motion on a given day, while experiencing the two FOV conditions on alternating days.

Familiarization and Training. During the familiarization phase of Experiment 1, each subject was allowed several minutes of unconstrained flight while being verbally coached. This was done to acquaint the subjects with control/display relationships as well as control stick sensitivities. Subjects then ran two static trials in which they attempted to maintain zero heading and an altitude of 30.5 meters in the presence of the simulated wind gusts, as described earlier. Subjects were then given two more trials with seat motion present. Half of the subjects were initially trained using the wide FOV display and the other half started with the narrow FOV display. Each step was then repeated under the remaining FOV condition to give the subjects equal experience with the two displays. At the end of each trial, subjects were presented the mean, standard deviation and root-mean-square errors for both heading and altitude.

For Experiment 2, the three naive subjects were trained in the manner stated above while the five returning subjects received no additional familiarization and training.

Data Collection. In both Experiment 1 and Experiment 2, subjects participated in a total of ten days of data collection. Each day, subjects experienced sixteen, 102-second trials -- eight with the seat on and eight with the seat off. The seat-on and seat-off conditions were ordered in an ABBA fashion, where A and B represented blocks of four trials. Subjects were presented either a wide or narrow FOV display on a given day.

Five of the six returned for Experiment 2; three naive subjects were added to Experiment 2 to bring the subject total to eight. None of the subjects were pilots.

Apparatus

Simulated Aircraft. A fighter-type aircraft was simulated on a Digital Equipment Corporation PDP 11/60 computer using simplified dynamics (Figure 1). The simulated aircraft was controlled using a side-mounted force-sensitive (isometric) control stick. Aircraft roll was regulated through lateral inputs while fore-aft inputs affected pitch. Roll and pitch gains were 35.23 and 3.43 degrees/second per newton-meter of torque, respectively, with a breakout torque of 0.21 newton-meters for both axes. The measured transport delay for the dynamic seat was 55 milliseconds while that of the visual display was 58 milliseconds; Johnson and Middendorf provide a detailed discussion of the delay verification procedures.¹³

Disturbances. The simulated wind gusts were implemented using spectral shaping filters, based on the Dryden gust model, driven by a sum-of-

sines approximation to white noise. The disturbance for each axis consisted of ten harmonically-unrelated sinusoids, ranging from 0.049 to 6.611 Hertz for the roll-rate disturbance and 0.029 to 5.403 Hertz for the pitch-rate disturbance, with the two spectra interleaved. The starting phase for each sinusoid was randomly assigned before each trial to eliminate the learning of disturbance patterns. The root-mean-square gust amplitude was 4.16 meters/second (moderate gusts) which resulted in root-mean-square amplitudes for the roll-rate and pitch-rate disturbances of 2.86 and 2.07 degrees/second, respectively.

Display. The visual scene consisted of a green grid on a black background depicting a perspective view of flat terrain as seen from the aircraft traveling at a speed of 128 meters/second (Figure 2). The size of each individual grid element was 30.5 by 122 meters.

The position and attitude of the simulated aircraft resulted from the control inputs of the subject and the simulated wind gusts. The resulting vehicle states were sent to a high-resolution raster-graphics system (Silicon

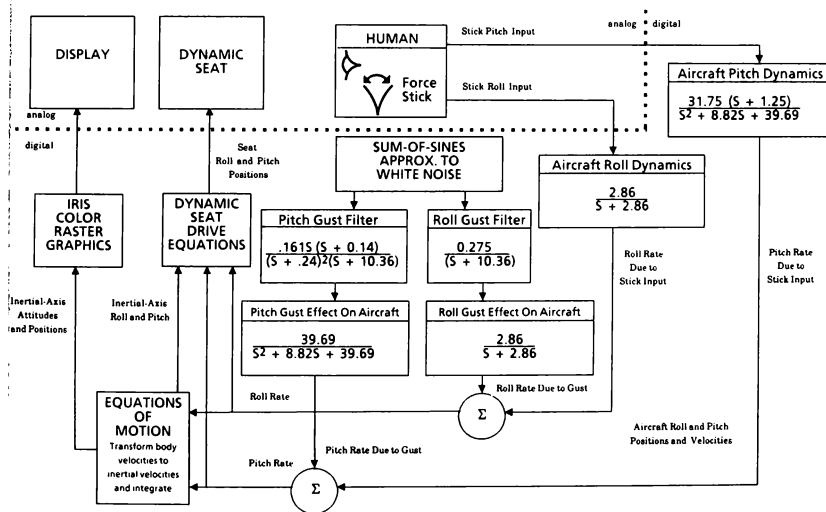


Figure 1. System Block Diagram

Results

Experiment 1

Previous research with this task⁴ suggested that there are differences in the quality of the altitude and attitude references provided by the visual display. In order to maintain the correct heading, subjects simply had to fly parallel to the longitudinal grid lines, which provide a good visual reference. Altitude regulation required that subjects keep the squares of the grid the same size that they were at the beginning of the trial; thus, the only reference was a mental image of the grid. This sometimes resulted in subjects maintaining an altitude which was offset from the desired 30.5 meters. To eliminate any differences in mean errors, roll, pitch, heading, and altitude standard deviations were chosen as the metrics for analysis. The standard deviations were averaged over the last eight trials for each subject under each of the four experimental conditions as an estimate of asymptotic performance.

The results are presented graphically in Figure 4. Repeated-measures analyses of variance were performed separately on the roll, pitch, heading, and altitude data. The FOV-by-dynamic-seat interaction was not significant ($p > .05$) in any of the analyses. Thus the graphs show only the main effects of dynamic seat cuing and FOV. Performance was significantly better for roll ($F(1,15) = 38.1, p < .0001$), pitch ($F(1,15) = 87.85, p < .0001$), heading ($F(1,15) = 38.86, p < .0001$), and altitude ($F(1,15) = 21.74, p < .0003$) when dynamic seat cuing was provided. The percentage of variance accounted for by the dynamic seat variable ranged from 22% to 75% in these analyses. The effect of FOV was not significant ($p > .05$) for any of the task components. However, performance tended to be better in the narrow FOV condition.

Experiment 2

Standard deviation measures were also chosen for analysis in this experiment. These data were averaged over the last eight trials for each subject under each of the four experimental conditions as an estimate of asymptotic performance.

The first analysis compared the performance of the five experienced and three naive subjects in

roll, pitch, heading, and altitude control. Since none of the tests showed a significant difference between the two groups, the data for all eight subjects were pooled for subsequent analyses. The pooled results are shown in Figure 5.

As in Experiment 1, the FOV-by-dynamic-seat interaction was not significant for any of the four task components. Dynamic seat motion again significantly improved performance in roll ($F(1,21) = 34.24, p < .0001$), pitch ($F(1,21) = 70.28, p < .0001$), heading ($F(1,21) = 41.44, p < .0001$), and altitude control ($F(1,21) = 17.51, p < .0004$). The percentage of variance accounted for by dynamic seat variable ranged from 6% to 40%.

Performance was significantly better with the narrow FOV condition for roll ($F(1,21) = 5.53, p < .0285$), pitch ($F(1,21) = 15.23, p < .0008$), and heading control ($F(1,21) = 7.04, p < .0149$). While the FOV variable had statistically significant effects, the percentage of variance accounted for was less than that of the motion variable. The values ranged from 5% to 8%.

Discussion

The results of Experiments 1 and 2 clearly support the prediction that dynamic seat cuing would improve performance under both wide and narrow FOV conditions. These data are consistent with the findings of Hosman and van der Vaart,¹⁰ but are counter to the common assumption that wide FOV displays reduce or eliminate the need for inertial motion systems.

The finding that performance was generally (Experiment 1) or even significantly (Experiment 2) better with the narrow FOV displays was completely unexpected. In fact, this finding in Experiment 1 led us to perform Experiment 2. The FOV effect in the first study might be accounted for by the superior brightness, contrast, and resolution of the monitor. These display characteristics would support more accurate estimation of heading and altitude errors under the narrow FOV condition. (We were aware of the potential confounding effect when designing Experiment 1, but we wanted to compare the old and new display systems, nevertheless.) However, the results of Experiment 2 do not support this explanation. In the second study,

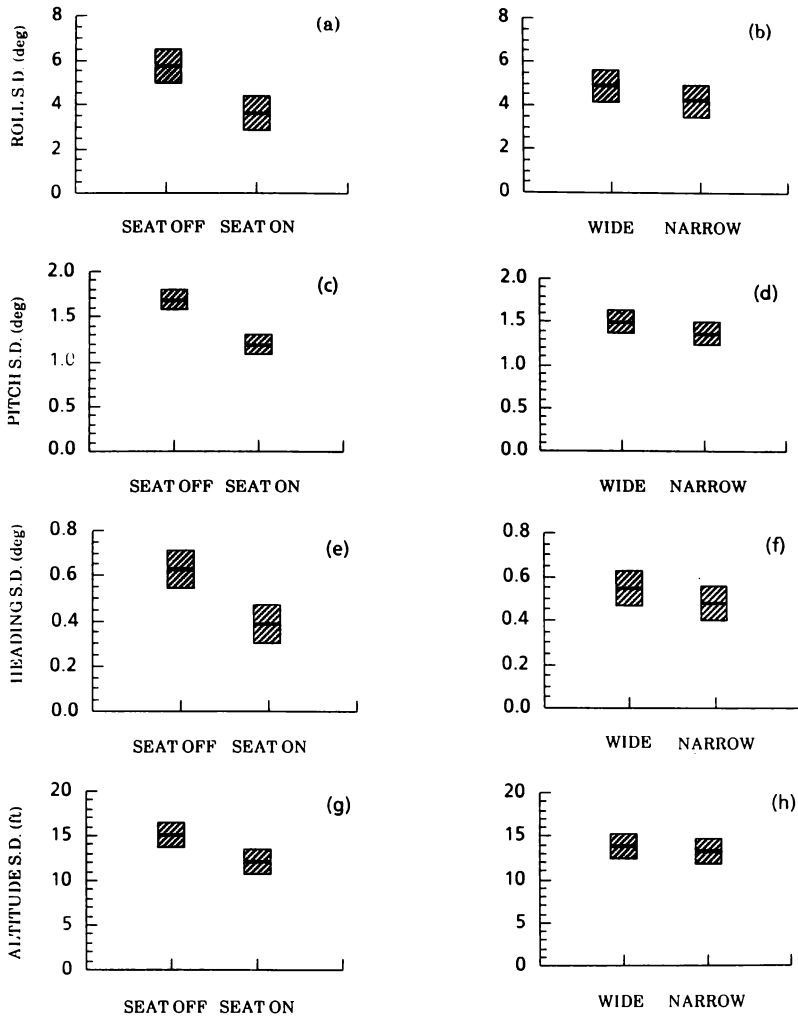


Figure 4. Plots a-h represent asymptotic group performance for Experiment 1. The centerlines of the boxes depict the group means under the specified condition. The top and bottom edges of the boxes represent the amount by which the means must be separated to be significantly different (95% simultaneous confidence intervals). Plots a,c,e and g compare performance with seat motion to that without seat motion; plots b,d,f and h compare performance with wide FOV to that with narrow FOV.

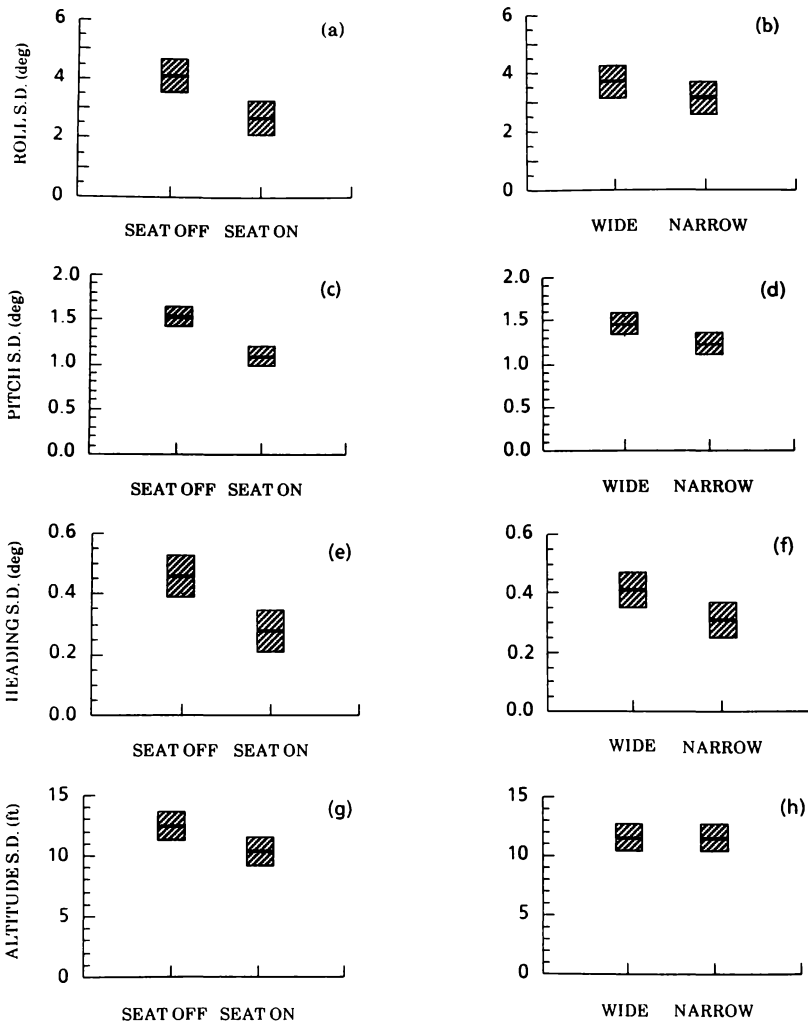


Figure 5. Plots a-h represent asymptotic group performance for Experiment 2. The centerlines of the boxes depict the group means under the specified condition. The top and bottom edges of the boxes represent the amount by which the means must be separated to be significantly different (95% simultaneous confidence intervals). Plots a,c,e and g compare performance with seat motion to that without seat motion; plots b,d,f and h compare performance with wide FOV to that with narrow FOV.

these display characteristics were equivalent under the two FOV conditions.

An alternative hypothesis is that some task characteristic led to better performance with the narrow FOV displays. Irish et al.,⁸ and Irish and Buckland⁹ found that FOV effects can be control-axis specific. In their studies, increasing the FOV generally improved roll, but not pitch performance. In the present study, performance in both axes was degraded by the larger FOV. The two Irish studies also found task-specific effects, showing better performance of a slow-flight task with a narrow FOV display. Unfortunately, the slow-flight task (aircraft control at near stall speed) and our task do not appear to be similar. Other conjectures are that the peripheral visual cues were not useful for this task and acted, in effect, as noise. Or perhaps the wider FOV display elicited more visual scanning by the subjects to judge altitude and attitude. Such an increase in scanning could lead to less accurate control.

The fact that performance degraded under the wide FOV condition limits the generalizations one can draw from this study. We have shown that dynamic seat cuing improves performance with a wide FOV display that either degrades or has no effect on control. We must also demonstrate that the seat is effective with a wide FOV display that improves performance. Then one could argue that the seat is still effective when it, and the peripheral cuing, are providing similar self-motion information. While Hosman and van der Vaart¹⁰ found this to be true with their task using platform motion, a similar demonstration with the dynamic seat must await future research.

References

1. McMillan, G.R., Martin, E.A., Flach, J.M., and Riccio, G.E. (1985). Advanced dynamic seats: An alternative to platform motion? *Proceedings of the 7th Interservice/Industry Training Equipment Conference*. (pp. 153-163). Arlington, VA: The American Defense Preparedness Association.
2. Flach, J.M., Riccio, G.E., McMillan, G.R., and Warren, R. (1986). Psychophysical methods for equating performance between alternative motion simulators. *Ergonomics*, 29, 1423-1438.
3. Osgood, R.K., Taylor, K., and McClurg, T. (1988). The dynamic seat as an angular motion cuing device. *Proceedings of the Human Factors Society - 32nd Annual Meeting*. (pp. 25-29). Santa Monica, CA: Human Factors Society.
4. Cress, J.D., McMillan, G.R., and Gilkey, M.J. (1989). The dynamic seat as an angular cuing device: Control of roll and pitch vs. the control of altitude and heading. *Proceedings of the Flight Simulation Technologies Conference*. (pp. 94-100). New York: American Institute of Aeronautics and Astronautics.
5. Gray, T.H. (1982). *Manual reversion flight control system for A-10 aircraft: Pilot performance and simulator cue effects* (AFHRL-TR-81-53). Brooks Air Force Base, TX: Air Force Human Resources Laboratory.
6. Westra, D.P., Simon, C.W., Collyer, S.C., and Chambers, W.C. (1982). *Simulator design features for carrier landing: I. Performance experiments* (NAVTRAEQUIPCEN-78-C-0060-7). Orlando, FL: Naval Training Equipment Center.
7. Westra, D.P., Sheppard, D.J., Jones, S.A., and Hettinger, L.J. (1987). *Simulator design features for helicopter shipboard landings: II. Performance experiments* (NTSC-TR87-041). Orlando, FL: Naval Training Systems Center.
8. Irish, P.A., Grunzke, P.M., Gray, T.H., and Waters, B.K. (1977). *The effects of system and environmental factors upon experienced pilot performance in the Advanced Simulator for Pilot Training* (AFHRL-TR-77-13). Brooks Air Force Base, TX: Air Force Human Resources Laboratory.
9. Irish, P.A., and Buckland, G.H. (1978). *Effects of platform motion, visual and g-seat factors upon experienced pilot performance in the flight simulator* (AFHRL-TR-78-9). Brooks Air Force Base, TX: Air Force Human Resources Laboratory.
10. Hosman, R.J.A.W., and van der Vaart, J.C. (1990). Motion perception and vehicle control. In R. Warren and A.H. Wertheim (Eds.), *Perception and Control of Self Motion*. Hillsdale, N.J.: Erlbaum.

11. Dichgans, J., and Brandt, T. (1978). Visual-vestibular interaction: Effects on self-motion perception and postural control. In R. Held, H. Leibowitz, and H.L. Teuber (Eds.), *Handbook of Sensory Physiology*, (Vol. VIII). Heidelberg: Springer Verlag.
12. Junker, A.M., and Price, D. (1976). Comparison between a peripheral display and motion information on human tracking about the roll axis. *Proceedings of the AIAA Visual and Motion Simulation Conference* (pp. 26-28). New York: American Institute of Aeronautics and Astronautics.
13. Johnson, W.V., and Middendorf, M.S. (1988). Simulator transport delay measurement using steady-state techniques. *Proceedings of the Flight Simulation Technologies Conference* (pp. 250-254). New York: American Institute of Aeronautics and Astronautics.
14. Kleinwaks, J.M. (1980). *Advanced low cost G-cuing system* (ALCOGS) (AFHRL-TR-79-62). Brooks Air Force Base, TX.: Air Force Human Resources Laboratory.

TIME DELAY COMPENSATION USING PERIPHERAL VISUAL CUES
IN AN AIRCRAFT SIMULATOR

Steven L. Lusk
Logicon Technical Services, Inc.
P.O. Box 317258
Dayton, OH 45431-7258

Cynthia D. Martin
University of Dayton Research Institute
300 College Park
Dayton, OH 45465

James D. Whiteley
Armstrong Aerospace Medical Research Laboratory
Wright-Patterson Air Force Base, OH 45433-6573

William V. Johnson
Systems Research Laboratories, Inc.
A Division Of Arvin/Calspan
Dayton, OH 45440

Abstract

The effects of simulator time delays on performance, control behavior and transfer of training were investigated using supplementary peripheral visual cuing. A disturbance-regulation task was used in which subjects were instructed to maintain a particular heading and altitude in the presence of pseudo-random wind gusts. The experiment took place in a fixed-base simulator with fighter-type dynamics. Delays used for the primary visual display were 67 and 300 ms. The peripheral horizon displays were either matched to the primary display, mismatched, or not present at all. Respectively, the four combinations of primary and peripheral display delays investigated were: (1) 67 ms / 67 ms, (2) 300 ms / 67 ms, (3) 300 ms / 300 ms, and (4) 300 ms / no peripheral display. Subjects trained in one of these four conditions for 50 trials, then "transferred" to one of two 67 ms conditions for the remaining 50 trials. At the end of training, subjects in the matched, minimal delay condition (67 ms / 67 ms) maintained heading significantly better ($p < .05$) than subjects in each of the other three conditions. Thus, delayed primary cues did degrade performance.

However, subjects with delayed primary cues and matched or mismatched peripheral cues (conditions 2 and 3) did not perform significantly better than subjects with delayed primary cues and no peripheral display (condition 4). This suggests that supplementary peripheral cuing, matched or mismatched, was not able to adequately compensate for the unresponsiveness of the simulated aircraft. There were no significant differences among the groups at transfer (trial 51).

Introduction

For flight simulation, *time delay* is defined basically as the unwanted computer-time required to transform manual control inputs into visual or motion display outputs. This delay is in addition to the normal response time of the simulated vehicle. Time delay research conducted at the Armstrong Aerospace Medical Research Laboratory (AAMRL), in the Human Systems Division, has been ongoing since 1984. The purpose of this research has been to measure the effects of simulator time delays on pilot control behavior, performance and transfer of training. This research is

largely in response to increasingly realistic visual-scene complexity, high-fidelity math models of aircraft dynamics, and cascaded simulation computer architectures, all of which contribute to the time delay problem.

Adding transport delay reduces the stability margin of a man-machine system, if the pilot does not adjust. Typical pilot adjustment involves a reduction in control aggressiveness (bandwidth), which can degrade the system's ability to follow inputs and/or null errors. Riccio, Cress and Johnson (1987) discovered a strong relationship between levels of delay (150 to 400 ms) and degraded operator performance for a turbulence regulation task. ⁸ Table 1 summarizes the effects of increasing

Type of aircraft	Increased error per 100 ms added delay	
	Hdg	Alt
Fighter	18%	13%
Transport	31%	17%

Table 1. Increased RMS error per 100 ms added delay, Fighter and Transport.

delay on heading and altitude maintenance error for both fighter-type and transport-type aircraft. They found that delays of 200 ms or greater were unacceptable (i.e., significant effects on control proficiency). These findings were consistent with in-flight research conducted by Calspan Corporation in the NT-33 variable stability aircraft. ¹

A variety of methods and techniques have been utilized to compensate for excessive transport delays; for a detailed discussion of the sources of transport delay and the various compensation techniques, the reader is directed to Gum and Martin (1987), McMillan (1990) and Johnson and Middendorf (1988). ^{3, 6, 5}

McMillan (1990) categorizes most of the conventional compensation techniques as either discrete or continuous. ⁶ Discrete techniques utilize predictor algorithms which typically

predict future aircraft position using some weighted combination of current aircraft velocity and acceleration. Continuous approaches typically utilize some type of filter(s) which generates phase lead to compensate for undesirable phase lag produced by transport delay.

In thinking about new ways to compensate for the effects of time delay, the work of Hosman and van der Vaart (1988) suggested some alternative approaches. ⁴ Their research examined control behavior and performance, using target-following and disturbance-compensation tasks. In one of their experiments, subjects were asked to estimate final roll-angle magnitude, while optimizing their response between accuracy and speed. The display conditions consisted of seven combinations of a central display, peripheral displays and cockpit motion. In short, they discovered that the addition of peripheral cuing to a primary display shortened response time (RT) by 60 ms. Likewise, the addition of cockpit motion shortened RT by 210 ms.

Merriken, Johnson, Cress and Riccio (1988) then examined whether such supplementary cuing devices might effectively aid pilots in compensating for a 200 ms transport delay in a primary visual display. ⁷ The task was to maintain heading and altitude in the presence of moderate turbulence. The supplementary cue information either preceded that of the primary display by 133 ms (i.e., mismatched) or was presented concurrently (i.e., matched). A dynamic seat, an attitude directional indicator (ADI) and peripheral horizon displays were the devices used for investigation; each provided attitude-only (i.e., pitch and roll) information. Thus, operators were not able to perform the task adequately using supplementary cues exclusively. When heading and altitude maintenance error of the groups having a supplementary cuing device was compared to those without, no significant differences were observed. However, the trends in heading maintenance error suggested that having a cuing device was beneficial whether matched or mismatched.

Results from Merriken, et. al. (1988) indicated that, of the three cuing devices, the

dynamic seat and the peripheral horizon displays comparably aided performance the most. In light of the Hosman, et. al. (1988) research, these findings are not surprising. Due to the relative cost-effectiveness and ease of addition to an existing simulator, peripheral horizon displays were chosen to conduct a follow-on quasi-transfer of training experiment.

Methodology

Subjects

Thirty-two college-age, non-pilot volunteers were paid to serve as subjects. All subjects were naive to the turbulence-regulation flight task.

Experimental Design

Subjects were assigned to 1 of 4 training conditions (three experimental groups and one control group, n = 8 per group). The independent variable was the training condition (four levels). Table 2 describes the experimental delays for both the primary and peripheral displays.

Group	N	TRAINING trials 1-50		TRANSFER trials 51-100	
		Primary Display Delay (ms)	Periph. Display Delay (ms)	Primary Display Delay (ms)	Periph. Display Delay (ms)
1	8	67	67	67	67
2	8	300	67	67	67
3	8	300	300	67	67
4	8	300	N/A	67	N/A

Table 2. Experimental conditions. Subjects trained in 1 of 4 assigned conditions, then transferred to 1 of 2 designated control conditions.

The dependent variables measured were root-mean-square (RMS) error scores for both heading and altitude.

Apparatus

A Digital Equipment Corporation Microvax II was used for both the aerodynamic math model calculations and data collection. The graphics for each display were generated by Silicon Graphics IRIS 3120s (one IRIS for the primary display and one for both of the peripheral displays). The monitors viewed by subjects were 19 inch diagonal, color, raster-scan monitors. The resolution was 768 lines by 1,024 pixels. A 60 Hz non-interlaced display mode was utilized.

Aircraft Dynamics. The dynamics were similar to those of a fighter-type aircraft flying at an altitude of 3048 m and a constant airspeed of 250 knots (128.5 m/s). The simulated aircraft was approximated by a first-order filter with a break frequency of 2.86 rad/sec in the roll axis and a second-order filter with a resonant frequency of 6.3 rad/sec in the pitch axis. Equations for the roll rate (1) and pitch rate (2) dynamics are as follows:

$$\frac{1}{\tau_R s + 1} \tag{1}$$

$$\frac{(\tau_{\theta 2} s + 1) \omega_{sp}^2}{s^2 + 2 \zeta_{sp} \omega_{sp} s + \omega_{sp}^2} \tag{2}$$

where:

- ω_{sp} - undamped natural frequency (short period) = 6.3 rad/sec
- ζ_{sp} - short period damping ratio = 0.7
- $\tau_{\theta 2}$ - first order pitch-mode time constant = 0.8 sec
- τ_R - first order roll-mode time constant = 0.35 sec

Coupling between the two axes was such that when a turn was initiated, an altitude loss occurred (unless pitch corrections were made). This same model has been used in this lab previously, not only for time delay research, but dynamic seat-cuing research as well. 8.7.2

Control Stick. The simulated aircraft was controlled using a side-mounted, isometric

force stick. The static control gains for the stick were set for a pitch rate response of 1.45 deg/sec (derived from 3 lb per g) and a roll rate response of 14.89 deg/sec when a one-pound force was applied 9.5 cm (3.75 inches) up from the base of the stick. The stick gains were set to record up to a 10.5 lb maximum force input in either the pitch or roll axis with a breakout force of 0.5 lb.

Control system. A simplified block diagram of the human-vehicle control system is shown in Figure 1.

test frequency. For a more detailed description of the delay-verification process, the reader is directed to Johnson, et al. (1988). 5

Disturbance. A Dryden-type wind gust model was used to perturb the aircraft pitch-rate and roll-rate during a trial. The model was driven by the sum of ten discrete, harmonically-unrelated sinusoids for each axis. This disturbance was an approximation of a Dryden gust model with an RMS gust amplitude of 13.65 ft/sec (moderate gusts). The frequencies of the spectra for the pitch and roll axis were

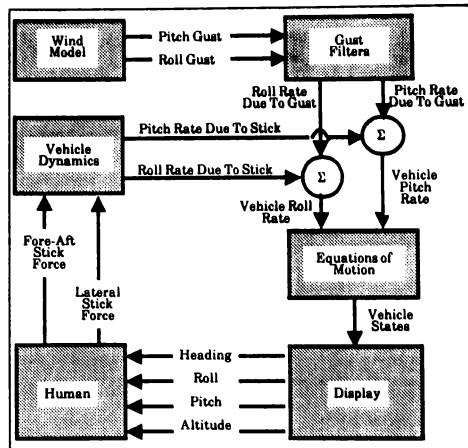


Figure 1. System Block Diagram of Man-Machine System.

Delay Verification. To measure the transport delay, several sinusoidal test frequencies were substituted for stick command inputs. A photocell was used to measure the simulator response on the visual display. The phase difference between the input and the output, measured by the photocell, was determined by a frequency analyzer (BAFCO model 916). The phase lag due to the aircraft dynamics was subtracted from the measured phase difference at each of the test frequencies. The transport delay was then calculated by dividing the adjusted phase difference by the

interleaved and separated by at least 1/4 octave from adjacent frequencies. These interleaved disturbance frequencies, for pitch and roll, allow for the separate investigation of control activity in each axis.

Displays. Figure 2 represents the placement of all three monitors. The monitors were situated such that the artificial horizon line (centered vertically in each display) for each monitor was aligned vertically with the subject's eye reference point. The viewing distance was 2 ft (0.61 m) as measured from the

center of each monitor to the bridge of the nose. The actual viewing area measured 11.5×15.5 inches (0.28×0.38 m) per screen, which provided a 26 degree high \times 35 degree wide

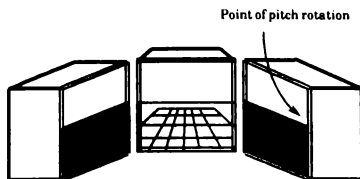


Figure 2. Experimental displays. The main display was a black grid overlayed onto a green surface; the "sky" was blue. Unlike the main display, the upper portion of the peripheral displays were white.

field of view for each monitor. Each monitor was placed on an adjustable table, allowing between-subject adjustments.

The **primary display** consisted of a blue sky and a black grid overlayed onto green (flat) terrain. The overall grid was 1,500 ft wide \times 6,000 ft long ($457.3 \text{ m} \times 1829.2 \text{ m}$); the grid cells were 100 ft wide \times 400 ft long ($30.5 \text{ m} \times 122.0 \text{ m}$). The grid was updated on the screen in a manner that provided the perception of forward motion, but in reality, the number of grid squares preceding the aircraft was constant (i.e., new grid cells were added at the horizon as the aircraft advanced).

The **peripheral monitors** faced the front of the subject at an angle of 72.4 degrees. The rear edge of the peripheral displays and the point of pitch rotation were coincident at eye level, such that the horizontal peripheral field of view encompassed the entire display. The peripheral displays provided attitude (pitch and roll) information only.

Procedure

Assignment of Subjects to Experimental Groups. Subjects completed 100 trials of an unstable dual-axis critical tracking task (CTT) prior to assignment to 1 of the 4 training conditions. Previous research has shown

statistically significant correlations between the CTT performance and (asymptotic) turbulence regulation performance. This finding permitted experimenters to place subjects into homogeneous groups, based on their CTT performance.^{7,8} Because most subjects reached a level of asymptotic performance during the final 20 trials, mean performance for those trials was used to rank-order and assign subjects to a condition.

Familiarization. Subjects were briefed on the first day of the experiment and given the opportunity to become familiar with the simulated aircraft. Subjects were coached through a set of pre-determined basic flight maneuvers until they were somewhat comfortable with controlling the simulated aircraft (i.e., limited or no control reversals, etc.); subjects were monitored very closely through this portion of the experiment. Subjects then completed five familiarization trials with the actual heading and altitude control task. Immediately following each trial, the mean error, standard deviation error, and RMS error was then displayed for heading and altitude. Scores were explained to the subjects as often as necessary, until each subject correctly evaluated each flight (trial) without assistance.

The Task. Subjects were required to maintain a heading parallel to the longitudinal grid lines, at an altitude of 100 ft (30.6 m), in the presence of pseudo-random roll-rate and pitch-rate disturbances. Subjects trained in their assigned condition for 50 trials (10 trials per day) and then transferred to a control condition for the remaining 50 trials (10 trials per day). The first control group (peripheral displays) remained in the minimal delay condition throughout the experiment. As indicated by Table 2, the fourth group transferred to a second control condition (minimal delay, no peripheral displays).

Monetary incentive. Merriken et al. (1988) noted a trend of increased RMS error scores during the final 10 trials ($41\text{--}50$) of the experiment, as subjects reached asymptotic performance. An inspection of each component of the RMS scores revealed that the increase in RMS error was caused primarily by an elevated

standard deviation. In an attempt to compensate for the subjects' apparent loss of motivation, a monetary incentive strategy was implemented. The plan was designed such that individual performance scores were used to calculate individual performance goals. This type of strategy encouraged subjects to do their best, but did not pressure them to do as well as the group.

Goals were established using averages of the standard deviation scores for both heading and altitude; payments were based on these goals. Merriken et al. (1988) observed that performance improved by approximately 30% over the first 40 trials and by approximately 10% over the next 10 trials. To maintain a performance level consistent with prior research, the following measures were taken: After subjects completed trials 1-20, their standard deviation scores were averaged. They were told that they would receive an extra dollar per hour if they improved this average score by 30% for the next 20 trials. Likewise, after subjects completed trials 21- 40, new standard deviation scores were averaged and they were informed that a 10% improvement of standard deviation scores over the next 10 trials (41-50) would result in payment of the bonus.

Results

Data Reduction.

First, the root-mean-square (RMS) heading and altitude data was (natural) log-transformed. A least-squares linear regression on the log data was used to generate predicted values which were plotted as learning curves for each subject. The predicted endpoints of the regression lines were then used for between-group comparisons at trials 1, 50, 51 and 100; trial 51 was the first transfer trial for experimental groups.^{8,7} The log-transformation method of data reduction was chosen as a means of achieving homogeneity of variance across levels of the independent variables (i.e., aircraft dynamics and/or transport delays) used repeatedly in our experiments; this method of data reduction is consistent with previous research conducted in this laboratory. For the following sections, numbers preceding the "/" correspond to the delay of the primary display; those following

the "/" correspond to the delay of the peripheral displays. One subject was discarded from the analysis as an outlier.

The means for Heading and altitude predicted RMS error are shown in **Figure 3 a & b**, respectively.

Performance Analyses

TRIAL 1: Results from the *general linear model* analysis of variance (ANOVA) in **Table 3** indicate that the training condition significantly affected heading maintenance [$F(3,27) = 4.40, p = .012$] for the first trial. Comparisons using a two-sample t-test showed

Trial	Dependent Variable	F-value (3, 27)	p-value
1	Heading Altitude	4.40 2.52	.012 .079
50	Heading Altitude	3.31 2.86	.035 .056
51	Heading Altitude	0.30 1.59	.825 .215
100	Heading Altitude	0.15 3.45	.926 .030

Table 3. ANOVA summary.

that condition 1 subjects maintained heading significantly better than both condition 3 subjects ($p = .0029$) and condition 4 subjects ($p = .0403$). In addition, condition 2 subjects were significantly better at heading maintenance ($p = .0305$) than condition 3 subjects.

TRIAL 50: Once more, the training condition significantly affected heading maintenance [$F(3,27) = 3.31, p = .035$]. Results of a two-sample t-test show that condition 1 subjects maintained heading significantly better than condition 2 ($p = .0068$), condition 3 ($p = .0131$) and condition 4 subjects ($p = .0357$).

TRIAL 51: As indicated by **Table 3**, there were no significant differences ($p < .05$) among the four groups when they initially transferred to the respective minimal delay control conditions.

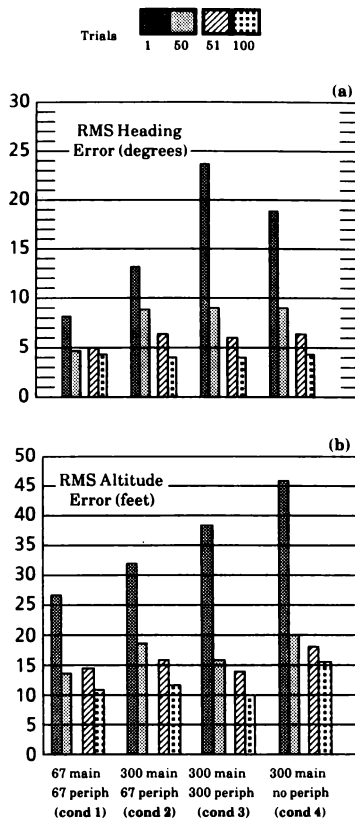


Figure 3 a & b. Heading and altitude mean predicted RMS error. Delays shown were training delays; subjects transferred to the minimal delay conditions. Significant pairwise differences were found for **heading** comparisons at trial 1: condition 1 was significantly better than 3 & 4; condition 2 was better than 3. Significant differences in heading also existed at trial 50: condition 1 was better than 2, 3 & 4. Significant pairwise differences for **altitude** control were found at trial 100: conditions 1 and 3 were better than 4, but not different from each other.

TRIAL 100: An inspection of Table 3 suggests that significant differences existed among the groups for altitude maintenance at trial 100 ($F(3, 27) = 3.45, p = .030$). Further inspection of the data with the two-sample t-test, indicates that condition 1 subjects were significantly better ($p = .0530$) at altitude maintenance than condition 4 subjects. Likewise, condition 3 subjects were better at altitude maintenance than condition 4 subjects ($p = .0233$).

Conclusions

In general, the 300 ms central display delay degraded heading performance. Specifically, control subjects in the 67 / 67 condition consistently exhibited better heading maintenance during training than all 300 ms condition subjects. Interestingly, the 300 / 67 subjects initially maintained heading significantly better ($p < .05$) than the 300 / 300 subjects. With this one exception, supplementary cuing (matched or mismatched) did not benefit subjects significantly in their attempt to compensate for the 300 ms central display delay. This suggests that the mismatched peripheral cuing was beneficial for the high-delay conditions early in training.

There appeared to be no significant differences in altitude maintenance between the groups throughout training, indicating that attitude control was not sensitive to delay. This also suggests that altitude control was not affected by the absence or presence of (matched or mismatched) peripheral visual displays.

Although 300 ms central display delay degraded heading performance throughout training, no significant differences were found among the groups upon transfer (trial 51). While subjects who trained in high-delay conditions (with or without supplementary cuing) were unable to maintain heading as well as control subjects at trial 50, when transferred to the same condition, all subjects performed equally well.

After completing 100 trials, subjects who trained in matched-delay conditions maintained altitude significantly better than subjects who trained without peripheral

displays. Rank ordering the mean scores for altitude (Figure 3-b) maintenance at trial 100 indicates that subjects who trained in matched-delay conditions were better than subjects who trained in the mismatched-delay condition and significantly better ($p < .05$) than subjects who received no peripheral supplementation at all. This finding is somewhat difficult to interpret and does not appear to have any simulator design implications.

Looking at the results from a cue-mismatch perspective, it is noteworthy that 233 ms central-peripheral mismatch had no significant effect on either heading or altitude maintenance, except very early in training. In this case (trial 1) performance was better with the mismatch.

Acknowledgments

The authors wish to thank Dr. Grant R. McMillan and Mr. Matthew S. Middendorf for their invaluable guidance and assistance throughout this entire process. We also thank Mr. Jeffrey L. Wood for the many hours he spent making sure that our data remained in a recoverable state and Mr. Michael J. Gilkey for his meticulous programming support. Thanks also to Mr. Charles D. Goodyear for his assistance with the statistical analysis. Finally, we thank Mr. Robert B. Jones for his technical assistance with the hardware configurations.

References

1. Bailey, R.E., Knotts, L.H., & Levison, W.H. (1987). *An investigation of time delay during in-flight and ground-based simulation* (Final Report No. 7205-16). Buffalo, NY: Calspan Corporation.
2. Cress, J.D., McMillan, G.R. and Gilkey, M.J. (1989). The dynamic seat as an angular cuing device: control of roll and pitch vs. the control of altitude and heading. *Proceedings of the AIAA Flight Simulation Technologies Conference* (No. 89-3336-CP). Washington, DC: American Institute of Aeronautics and Astronautics.
3. Gum, D.R., and Martin, E.A. (1987). *The flight simulator time delay problem*. Presented at the AIAA Flight Simulation Technology Conference. Washington, DC: American Institute of Aeronautics and Astronautics.
4. Hosman, R.J.A.W. and van der Vaart, J.C. (1988). Visual-vestibular interaction in pilot's perception of aircraft or simulator motion. *Proceedings of AIAA Flight Simulation Technology Conference* (No. 88-4622-CP). Washington, DC: American Institute of Aeronautics and Astronautics.
5. Johnson, W.V. and Middendorf, M.S. (1988). Simulator transport delay measurement using steady-state techniques. *Proceedings of AIAA Flight Simulation Technology Conference*. Washington, DC: American Institute of Aeronautics and Astronautics.
6. McMillan, G.R. (1990). *Cue integration and synchronization*. Armstrong Aerospace Medical Research Laboratory, Wright-Patterson Air Force Base, OH.
7. Merriken, M.S., Johnson, W.V., Cress, J.D. and Riccio, G.E. (1988). Time delay compensation using supplementary cues in aircraft simulator systems. *Proceedings of the AIAA Flight Simulation Technologies Conference* (No. 88-4626-CP), Washington, DC: American Institute of Aeronautics and Astronautics.
8. Riccio, G.E., Cress, J.D., and Johnson, W.V. (1987). The effects of simulator delays on the acquisition of flight control skills: control of heading and altitude. *Proceedings of the Human Factors Society - 31st Annual Meeting*, 2, 1286-1290.

WHEN IN-FLIGHT SIMULATION IS NECESSARY

Philip A. Reynolds*
 Valerie J. Gawron**
 Calspan Corporation
 P.O. Box 400
 Buffalo, NY 14225

Abstract

As aircraft systems have become more complex, designers have depended more and more on ground simulation to optimize the aircraft's design. But when are the answers from ground simulators misleading or even erroneous? When is in-flight simulation necessary? Previous in-flight simulation programs and research experiments are reviewed and a set of guidelines for answering these questions presented.

Previous Research

1: *Supersonic Transport Simulation (1972)*

The FAA tested a detailed approach and landing model of the unaugmented Concorde in the United States Air Force (USAF) Total In-Flight Simulator (TIFS; see Figure 1) and the NASA/Ames Flight Simulator for Advanced Aircraft (FSAA). This experiment, to define the Cooper-Harper Level 2/3 boundary as center of gravity is moved aft, was the first to attempt to compare complete six degree-of-freedom (DOF) ground simulation with in-flight simulation and with actual aircraft performance. Both simulators tended to predict the 6.5 boundary at the same value of time to double amplitude, although there was considerable scatter in the data.¹¹ Concorde pilots were enthusiastic



Figure 1

about how well in-flight simulation (IFS) produced the Concorde handling qualities. They did report, however, that the ground effect was too abrupt and too large in both the TIFS and the FSAA.

2: *YF-16 (1973)*

In the evolution that yielded the USAF's primary lightweight fighter, the F-16 underwent a development process for its sidestick controller and flight-control system (FCS). This process included a fixed-base ground simulator for pilot-in-the-loop tests and evaluation prior to first flight. In addition a short IFS was conducted in the USAF NT-33A Variable Stability Aircraft (see Figure 2).

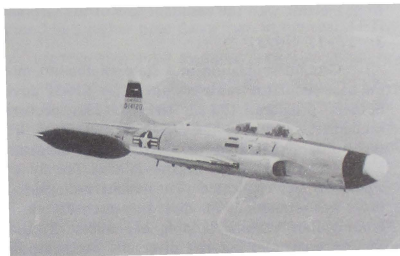


Figure 2

The F-16 fly-by-wire FCS and fixed sidestick were simulated in landing, formation, and engine-failure flameout approaches. The results of the IFS indicated that: 1) roll response was too sensitive and would lead to roll ratchet and roll Pilot-Induced Oscillation (PIO) and 2) there was a pitch bobble (or PIO) tendency.⁵

The oil embargo halted evaluations after one week and the NT-33A was sent home. However, the YF-16 roll gearing was reduced by factor of two as a result of the IFS evaluations. Soon after, an inadvertent YF-16 first flight resulted from roll PIO encountered during high-speed taxi test. Control surface rate limiting was also a factor in the PIO.

* Principal Aeronautical Engineer, Flight Research Department; Member AIAA
 ** Principal Human Factors Engineer, Flight Research Department; Member AIAA

The aircraft's left and right stabilizer and right wing contacted the runway. The General Dynamics pilot saved the aircraft by taking off.

The NT-33A returned to Edwards Air Force Base for additional F-16 IFS. As a result, the roll gearing was further reduced in YF-16 and, subsequently, the "first" flight was successfully made. Ultimately, the YF-16 roll gearing was considerably reduced from the value selected as optimum in ground simulation.

During the YF-16 development testing, the NT-33A simulation was used to train and check out each pilot before flying the YF-16. Usually, three training flights were required. Emphasis was placed on evaluating the sidestick-controlled handling qualities in landings and takeoffs and on simulated flameout landings to lake beds and runways. During YF-16 testing, several engine roll-backs to idle were experienced which required landing without power. One Tactical Air Command (TAC) pilot assigned to the test force had this happen on his first flight in the YF-16. He made a successful landing, stating that his confidence in being able to land the F-16 was a result of his NT-33A training. This training kept him from ejecting with the resultant loss of the airplane.

3: YF-17 (1973)

Ground simulation in a state-of-the-art moving base simulator was used during the YF-17 development program. IFS in the NT-33A occurred immediately prior to the YF-17's first flight. The flight phases simulated included landing, formation, air-to-air tracking, and engine-failure. The fly-by-wire FCS was replicated. The results indicated: 1) there was a tendency for divergent pitch PIO during landing flare making landing impossible, 2) there was an oscillatory but not divergent pitch response during tracking and formation, and 3) neither were reported in ground simulation.

A series of rapid changes and evaluations were performed in the next two weeks to improve the YF-17's FCS.⁵ First the YF-17 pitch command pre-filter was easily bypassed in the NT-33A by throwing a switch. No PIO occurred with the pre-filter removed but then the pitch response was too abrupt. Second, mathematical analysis predicted that a properly modified (less filtering; first order rather than second order) pre-filter would cure the problem. The NT-33A was so modified (overnight) to simulate that pre-filter, and subsequent IFS evaluations confirmed the predictions. Third, the YF-17 was modified accordingly but only when the landing gear was down. When the gear was retracted, the control system reverted to the original pre-filter. The YF-17 was then flown successfully on its first flight with good handling reported with gear

down. When the gear was retracted (and the original pre-filter inserted), the pitch response was oscillatory in tight tasks as predicted by the NT-33A IFS. After several confirming flights, the new pre-filter was incorporated throughout the flight envelope.

4: Advanced Supersonic Transport (1977)

NASA performed IFS in the TIFS following development of the flight control system on a fixed-base ground simulator at NASA/Langley. The overall rating in the ground simulator was 2. In TIFS it was 7 to 8¹². The reason for the poor flying qualities ratings during IFS was the high side acceleration at the cockpit due to rolling about the flight path 36 feet below in the approach condition. Side acceleration at the cockpit had been displayed to the pilot in the ground simulator but the ball was driven with side acceleration at the center of gravity. The flight control designers' attention had not been attracted to time histories of the side acceleration. However, the motion cue was impossible to overlook in TIFS. The roll axis had been augmented with roll rate feedback and the sensitive roll command gain chosen yielded almost 0.1 g sideways acceleration per pound input. A second series of TIFS tests in 1978 yielded better flying qualities with a less sensitive roll axis.

5: F-18 (1978)

There was extensive ground simulation during the F-18 development program. IFS was performed in the NT-33A prior to first flight. This was the first IFS of a digital flight-control system. The primary emphasis was on Navy field carrier landings (but with shallower approach angle due to NT-33A landing gear strength limits). Evaluation results⁹ indicated: 1) a tendency for divergent roll PIO to occur just prior to touchdown and 2) the sideslip rate estimator (to provide yaw damping) was incorrectly mechanized in the F-18 flight control computer leading to very poor flying qualities. NT-33A evaluation showed that the roll PIO tendency could be improved by reducing the transport time delay in roll axis and by reducing the roll gearing. Subsequently the digital sampling rate was increased, the roll gearing was reduced, and the sideslip rate estimator algorithm was modified prior to first flight. There was significant improvement in the flying qualities.

6: Space Shuttle (1972 to 1985)

NASA requested IFS late in the Space Shuttle development program and funded five evaluation programs from 1972 through 1985. Each evaluation focused on the approach and landing task, including approach and pre-flare segments. TIFS was the IFS vehicle used throughout these evaluation programs. Early programs examined the basic shuttle aerody-

namics, handling, control laws, and rotational hand controller with variations made to assess the effects of uncertainties in the expected characteristics.

Mathematical analysis of the mature shuttle design predicted pitch PIO tendencies although ground simulation did not support this conclusion. TIFS was not used to test the mature configuration. When pitch PIOs were experienced during landing on Approach and Landing Test 5, a TIFS simulation program was performed. That simulation reproduced the PIO tendency. TIFS was used to test and optimize a PIO-suppressor device that became a part of the operational Shuttle vehicles¹³.

Like the YF-16 takeoff incident, rate limiting was a factor in the PIO. The Shuttle has a priority rate limiting scheme on the elevons which determines how much pitch and roll control is available when the surfaces are rate limited. The pilot has reduced authority in one axis when there is excessive activity in the other axis. Predictions obtained from ground simulations of the control activity in a typical landing were not reliable.

NASA used the early TIFS simulations to give the opportunity to "fly the unpowered Shuttle to landing" with the real-world cues and environment, adding confidence that the job could really be done. These simulations included landings in actual instrument weather conditions, which have not yet been done with the real shuttle. IFS (both the TIFS and the Shuttle Training Airplanes) enabled the pilots to develop control techniques that allow them to overcome the shortcomings in the Shuttle flying qualities.

7: *Advanced Subsonic Transport (1980)*

The Dutch National Aerospace Laboratory (NLR) conducted a moving-base ground-simulator program to examine fly-by-wire flight control on a derivative of the F-28. NLR then used TIFS to verify or "calibrate" the results, particularly in the flare and touchdown portion of the landing task. Pilot rating variations with changes in the pitch response parameters were generally the same on the ground and in flight, although touchdown sink rates averaged 4 fps on the ground simulator and 2 fps in TIFS. Speed control was a problem on both the ground and in flight, but pilot complaints were more numerous and louder in flight. Direct lift control gain increases on the ground produced a strongly favorable trend in pilot rating. In flight, this trend did not appear and even showed a mild reversal due to abrupt cockpit motions¹⁰.

8: *Advanced Fighter Technology Integrator (AFTI/F-16) (1981)*

No IFS was planned during the AFTI/F-16 development program. All development was done

using a ground simulator. Prior to first flight, FDL requested and funded an NT-33A simulation. The NT-33A IFS focused on the landing phase with simulation of the primary FCS and the reconfiguration (backup) FCS modes. The results indicated that the pitch rate reconfiguration mode was un-flyable for landing. The FCS was subsequently extensively revised⁸. Further, the yaw rate reconfiguration was modified as a result of IFS.

To settle the considerable debate as to correct pitch time delay to represent the real AFTI, an additional simulation was performed using the NT-33A. In this IFS, the NT-33A simulated the F-16 with the new pitch-rate control system that was being flown at Edwards AFB. One USAF test pilot flew the F-16 and NT-33A simulation and stated that he couldn't see any difference between the two aircraft, giving credibility to IFS.

When the real AFTI/F-16 flew, the results had good correspondence with IFS except in one mode: pitch normal digital. Investigation showed that a pre-filter had been incorrectly interchanged with a nonlinear gearing box in the NT-33A during the third simulation (done to permit in-flight variation of time delay parameter). This difference resulted in a higher time delay than in the real AFTI.

9: *VISTOL Test Pilot Training (1983)*

The Navy Test Pilot School used the X-22A (see Figure 3) as a training tool to acquaint students with thrust-vectoring flight control. The X-22A was operated as a fixed-base ground simulator using the evaluation pilot's feel system, Head-Up Display (HUD), and other displays driven by computer. Then, the same configurations were evaluated in-flight in the X-22A. Most pilots learned to perform the approach task quite well in the ground simulator with the exception of the rate-command flight-con-

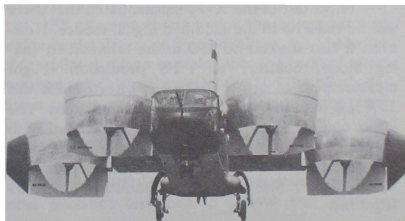


Figure 3

tol configurations. They had so much difficulty that until the Spring of 1983, the rate systems were not attempted in flight. The students from the Spring 1983 class decided to test the rate systems in flight and found to everyone's surprise that they could achieve acceptable performance¹.

10: Command Flight Path Display (1983)

A pictorial-type pathway display was tested in TIFS. As part of the checkout procedure, the display was flown on the ground using the TIFS evaluation cockpit and the TIFS equations of motion in a computer. Both increased Dutch roll damping and reduced yaw due to ailerons were used on the ground to reduce the lateral excursions from the pathway. In flight the display was easy to fly and the Dutch roll damping increment was not needed².

11: X-29 (1984)

IFS was not used for the X-29 development program initially. The FCS was developed in a fixed-base ground simulation and also tested in a motion-based simulation. However, IFS was requested and funded by FDL when the FCS development was mature. TIFS was used as in-flight simulator for X-29 because it could completely match the canard effects with its full six DOF simulation capability.

IFS evaluations uncovered roll PIO problems in landing in all three FCS modes. The problems were worst in the normal and backup digital modes. IFS evaluations also showed a tendency for the X-29 to float during landing. These problems were not present in the ground simulators. Further IFS evaluations demonstrated that reducing the roll gearing by a factor of two eliminated the PIO problems and provided good roll handling. This change was made to X-29 FCS digital modes prior to first flight; the analog mode was left unchanged because of the high cost of modification at that point.

Flight test of the X-29 substantiated that there was no roll PIO in the modified digital modes. However, it also showed no PIO in the unmodified analog mode, contrary to TIFS prediction. Flight measurement of actual X-29 roll effectiveness was 30% less than the value used in all simulations. The lower control effectiveness had an effect similar to lowering the control gearing as had been done to fix the digital modes. The floating tendency predicted by in-flight simulation was substantiated by X-29 flight test.

12: Israel's Lavi Fighter (1984, 1985)

The initial control law development of the Lavi Fighter was performed in ground simulation. The resultant FCS was evaluated in IFS using the NT-33A.

The IFS results caused significant modifications to be made to control laws. Some command gains were reduced. Further, the IFS results enabled engineers to better interpret the subsequent ground simulation results. The first flight was flown in early 1987. The Lavi pilots indicated that the aircraft's flying qualities were excellent right from the start of flight testing.

13: Simulation Validation Experiment (1985)

Five configurations based on two sets of dynamics with time delay as the primary variable were flown in the NT-33A and the NASA/Ames Vertical Motion Simulator (VMS). The purpose of this experiment was to evaluate and compare configurations with PIO tendencies in an in-flight and a moving ground-base simulator. The same configurations were flown in the NT-33A and the VMS in December 1985 by the same pilots performing the same task (precision visual landing from an offset).

The best configuration was rated 3.5 on the average in flight and on the ground (eight data points in flight and six in the VMS using three pilots in each). The standard deviation was 1.3 in flight and 0.5 in the VMS.

The worst configuration (it was chosen to be the YF-17 data point of case 3) was rated 8.2 on the average in flight and 6.5 in the VMS (six data points and three pilots in each). The standard deviation was 1.2 in flight and 1.8 in the VMS. Four of the six ratings in flight were 8 or worse with the same severe PIO tendencies as were observed twelve years previously. Only one rating in the VMS was 8 or worse.

The NASA/Ames conclusions, based on preliminary evaluation of the data and pilot comments, were: 1) ratings of "PIO prone" configurations are extremely sensitive to task and to pilot control technique (aggressiveness), 2) in both the IFS and the VMS (for the standard lateral-offset approach, precision landing task), some pilots observed serious PIO tendencies while others did not, 3) tasks which require urgent or aggressive pilot control inputs more effectively expose latent flying qualities deficiencies, and 4) there is a need for a national program to acquire generic data to support military flying-qualities specification requirements.

14: Time Delay Study (1986)

An in-flight experiment was performed in the NT-33A to investigate the effects of time delay on manual flight control and flying qualities. The in-flight time delay data were generated with full fidelity, unlimited range of motion cues. Using the same cockpit and a digital aerodynamic simulation, the in-flight experiment was completely replicated as a fixed-base ground simulation. For almost every

value of added time delay, tracking error and the pilot rating were poorer in the ground-simulator mode than in the in-flight simulator mode.⁴

Generalization Issues

Bise, Luebke, Lehman, and Masters⁸ identified several issues associated with generalizing the results of ground simulators to in-flight tests:

- 1) the pilots' perceived risk in a ground simulator is low relative to real flight;
- 2) systems in ground simulators tend to function more ideally than in flight (e.g., sensors are fully functional when activated; radio communications are crystal clear);
- 3) ground-base simulators either have no motion cues or motion cues which are distorted by washout; and
- 4) domed simulators have objects displayed within optical infinity; aircraft have objects displayed at the actual distance including beyond optical infinity.

Reynolds⁸ identified other issues that must be addressed in comparing ground simulation and in-flight test results:

- 1) ground simulations that use strong peripheral rate cues in place of rolling the platform have caused disorientation of subjects, which lasts several hours after the simulation session;
- 2) state-of-the-art visual scene generators are at best an order of magnitude less bright than ordinary sunlight;
- 3) the scene detail in a ground simulator is not as great as the real world;
- 4) a real flight cannot be stopped in mid-motion as a simulation scene can be; and
- 5) ground simulators have added time delays between pilot input and the corresponding visual and/or motion cues; such delays have been shown to affect both workload and performance⁴.

Guidelines for the use of in-flight test versus ground-based simulation are summarized in Table 1.

Guidelines

The argument for IFS is at the same time persuasive and elusive. It always involves resources available and usually involves management policies. Our assertion is that there are objective guidelines for making engineering and management decisions

Table 1
CREW STATION EVALUATIONS
APPROPRIATE FOR GROUND SIMULATIONS
AND IN-FLIGHT TESTING

What Can Be Done in Ground Simulation?	What Must Be Done in Flight Testing?
Evaluating crew station form/fit at 1 g	Evaluating crew station form/fit over the aircraft's full range of g's
Evaluating Pilot Vehicle interface procedures not requiring high fidelity visual or motion cues	Evaluating PVI procedures requiring high fidelity visual cues (e.g., aerial refueling) or motion cues (e.g., aggressive landing)
Identifying the sensor resolution required to meet mission objectives	Identifying the actual sensor resolution and how it impacts meeting the mission requirements
Evaluating display formats	Evaluating sunlight readability and vibration readability of display formats
Evaluating control placement and shape	Evaluating controller role in handling qualities
Evaluating crew station integration in a benign environment	Evaluating crew station integration in vibration, pressurized cockpit, extreme temperatures, and direct sunlight

on the use of in-flight simulation. These guidelines and their supporting evidence are given in Table 2.

Caveats

IFS can be feared like the ancient Oracle. If it is wrong, how do you defeat it? Yet it can be wrong if the model is wrong or the evaluations are not done carefully. It puts the evaluation pilot on the spot and, at the same time, gives the pilot a more powerful voice in the decisionmaking.

Simulation always involves some elements of make-believe - otherwise it would not be simulation. IFS attempts to maximize the components of reality in simulation of flying qualities, i.e., behavior of the pilot-airplane combination, the level of effort required to achieve that behavior and the pilot's decision about the relationship between the two. Ground simulation attempts to do the same thing - sometimes quite successfully. The important question is when is the answer ground simulation gives not to be trusted? We hope this paper has provided some guidance on when to believe and when to question.

Table 2
GUIDELINES FOR IN-FLIGHT SIMULATION

Condition Suggesting Use of In-Flight Simulation	Supporting Evidence
High gain task	YF-16 landing, formation and engine-failure flameout approaches YF-17 landing, formation, air-to-air tracking, and engine failure Space Shuttle approach and landing Simulation validation experiment difficult landing
New controller	YF-16 side controller Space Shuttle rotational hand controller
Side acceleration	Advanced Supersonic Transport
New FCS	F-18 digital FCS AFTI/F-16 primary and reconfiguration FCS modes V/STOL Test Pilot Training rate-command flight-control configuration Israel's Lavi Fighter X-29 normal and backup digital FCS
Abrupt Cockpit Motion	Advanced Subsonic Transport
New flight director	Command Flight Path Display
Time-delay greater than 150 msec	Time delay study
Rate limiting	YF-16, FCS Space Shuttle FCS

Lessons Learned

Lesson 1, control sensitivities, such as roll rate or g's per pound of stick force, are frequently chosen too light when the actual motion is appreciably attenuated or not present. Lesson 2, the model can usually be simplified for flying-qualities tests, but simplifications must be justified. Lesson 3, important model parameters should be varied to determine their flying qualities sensitivity, especially if changes in the FCS are at issue late in the development cycle. With augmented vehicles, control surface effectiveness becomes more important. It can dominate flying qualities. Lesson 4, feel systems must accurately reflect what is in the real aircraft. They are not a detail to be treated with gross approximation. Lesson 5,

displays need not be exact replicas but they should include all the information important to the task presented in a similar manner.

Lesson 6, excessive simulation time delay can cause PIOs or degraded pilot ratings. Use of feed forward techniques can reduce simulation time delay, in some cases to zero. Visual and motion cues should be synchronized. Lesson 7, the task should be defined in detail. If the real operational situation includes the likelihood of tight, closed-loop control, this should be part of the simulator task and difficult for the pilot to avoid by subtly changing the task. Lesson 8, pilot background and characteristics as a controller should be part of the experimental data. The pilots should receive the same briefing and the briefing should emphasize the task, the rating scale use, and the comment card. Lesson 9, expect scatter in pilot ratings. Use repeat evaluations and other evaluation pilots to determine a consensus. Use the comment card to obtain insight on reasons for the rating, special control techniques developed, learning effects, and pilot confidence in the rating. Lesson 10, use the pilot in a real-time development mode. If some characteristic is objectionable, change it on the spot and get the reaction.

Lesson 11, engineers should look at time histories of model motion (and cab motion, if present). They should remember that the pilot will feel complete cockpit motions in flight - not washed out motions. They should become aware of how transient roll rates of 10 degrees per second and side accelerations of 0.05 g feel when doing a landing approach. All six DOF should be checked. Lesson 12, excessive control activity leading to rate limiting should be examined in an in-flight simulator matching the actual task as closely as possible.

References

1. Beilman, J.L., "Indoctrination of Navy Test Pilots to Vectored Thrust Flight in the X-22A In-Flight Simulator," AIAA Paper 83-1076, AIAA Flight Simulation Technologies Conference, June 1983.
2. Bise, M.E., Luebke, L.M., Lehman, E.F., and Masters, R.M., "Design Methodology for an In-Flight Pilot Performance and Workload Evaluation Instrumentation System," Wright-Patterson Air Force Base, OH: Human Systems Division (OL-AC), December 1987.
3. Dittenhauser, J.N., Eulrich, B.J., and Reynolds, P.A., "Command Flight Path Display (CFPD) Sensor Software Development and Overall System Hardware Integration in the NC-131H (TIFS) In-Flight Simulator,"

- Calspan Report Number 6645-F-12, December 1983.
4. Gawron, V.J., Bailey, R.E., Knotts, L.H. and McMillan, G.R., "Comparison of Time Delay During In-Flight and Ground Simulation," Proceedings of the 33rd Annual Meeting of the Human Factors Society, 120-123, 1989.
 5. Hall, G.W. and Harper, R.P., "In-Flight Simulation of the Lightweight Fighters," AIAA Paper 75-985, AIAA Aircraft System and Technology Meeting, August 1975.
 6. Harrington, W.W., Van Vliet, B.W., and Unfried, F.E., "Assessment of Advanced Fighter Power Approach Simulations," AIAA Paper No. 83-0141, AIAA Aerospace Sciences Meeting; January 1983.
 7. Knotts, L.H. and Priest, J.E., "NT-33A In-Flight Simulation of DIANA, Session 2 — Landing/Approach Evaluation," Calspan Report Number 7205-F-11, December 1985.
 8. Reynolds, P.A., "Flying Qualities Evaluation Using Ground and Airborne Simulation," Total In-Flight Simulator Technical Memorandum Number 1216, September 2, 1986.
 9. Smith, R.E., "Evaluation of F-18A Approach and Landing Flying Qualities Using an In-Flight Simulator," Calspan Report Number 6241-F-1, February 1979.
 10. Van Gool, M.F.C. and Weingarten, N.C., "Comparison of Low-Speed Handling Qualities in Ground-Based and In-Flight Simulator Tests," AIAA Paper 81-2479, AIAA Flight Test Conference, November 1981.
 11. Wasserman, R. and Mitchell, J.F., "In Flight Simulation of Minimum Longitudinal Stability for Large Delta-Wing Transports in Landing Approach and Touchdown," AFFDL TR-72-143, February 1973.
 12. Weingarten, N.C., "An Investigation of Low Speed Lateral Acceleration Characteristics of Supersonic Cruise Transports Utilizing the Total In-Flight Simulator (TIFS)," Calspan Report Number 6241-F-2, March 1979.
 13. Weingarten, N.C., "In-Flight Simulation of the Space Shuttle Orbiter During Landing Approach and Touchdown in the Total In-Flight Simulator (TIFS)," Calspan Report Number 6339-F-1, September 1978.

THE AERODYNAMIC EFFECT OF HEAVY RAIN ON AIRPLANE PERFORMANCE

Dan D. Vicroy*

NASA Langley Research Center, Hampton, Virginia

Abstract

The National Aeronautics and Space Administration has been conducting a series of tests to determine the effect of heavy rain on airfoil aerodynamics. The results of these tests have shown that heavy rain can significantly increase drag as well as decrease lift and stall angle of attack. This paper describes a recent effort to use the heavy rain airfoil data to determine the aerodynamic effect on a conventional twin-jet transport. The paper reports on the method used to model the heavy rain aerodynamic effect and the resulting performance degradation. The heavy rain performance effect is presented in terms of the diminished climb performance associated with increasing rain rates. The effect of heavy rain on the airplane's ability to escape a performance-limiting wind shear is illustrated through a numerical simulation of a wet microburst encounter. The results of this paper accentuate the need for further testing to determine scaling relationships and flow mechanics, and the full configuration three-dimensional effects of heavy rain.

Symbols

b	wing span
c	chord
\bar{c}	mean aerodynamic chord
cd	section drag coefficient
cl	section lift coefficient
CD	total drag coefficient
CL	total lift coefficient
F	wind shear hazard index
g	gravitational acceleration
Sl	span load coefficient $\left(\frac{c \cdot cl}{\bar{c} \cdot CL}\right)$
V	true airspeed
W_h	vertical wind component, updraft positive
W_x	rate of change of horizontal wind component, tailwind positive
α	angle of attack

Introduction

Wind shear is considered by many in the aviation industry to be one of their major safety issues. Numerous accidents and incidents have occurred which were attributed to low-

altitude wind shear. Many of these were accompanied by rain, some of which was classified as intense or heavy rain. The effects of rain on airplane performance under normal operating conditions, are generally not considered to be significant. However, in a performance limiting wind shear, the airplane may require large angles of attack to maintain a safe altitude. The performance influence of heavy rain under these conditions may be considerable.¹

The objective of this study was to estimate and characterize the effect of heavy rain on the performance of a conventional twin-jet transport. This required the development of an aerodynamic model of the airplane which included the rain effect. This paper will discuss the development of such a model based on the results of a series of tests which measured the effect of heavy rain on airfoil aerodynamics. The paper will initially summarize the results of the heavy rain airfoil tests. The method used to develop the heavy rain aerodynamic model will then be discussed. This will be followed by a performance analysis with the heavy rain model. The final section illustrates the effect of heavy rain on the airplane's ability to escape a performance limiting wind shear through a numerical simulation of a wet microburst encounter.

Heavy Rain Airfoil Tests

In an effort to measure the effect of heavy rain on airfoil aerodynamics a series of tests was conducted in the NASA Langley 14- by 22-Foot Subsonic Tunnel on a cambered airfoil section representative of the type used on commercial transport aircraft.² The section model had a rectangular planform with a 2.5 foot chord and a 8 foot span, mounted between rectangular end plates. The airfoil section was a NACA 64-210. The model was tested in a cruise and a high lift configuration. The high lift configuration had a leading edge slat and a trailing edge double slotted flap as shown in figure 1. The rain was simulated by injecting a water spray horizontally toward the model from a rain spray system mounted upstream. The spray system could produce water mass concentrations from 16 to 46 g/m³. A photograph of the wind tunnel model immersed in the water spray is shown in figure 2.

The results of the wind tunnel tests showed a reduction in maximum lift capability and stall angle of attack and a corresponding increase in drag with increasing concentrations of liquid water. The tests also showed that the cruise configuration was less sensitive than the high lift configuration to the rain environment. The lift and drag coefficient data obtained from the tunnel test for the cruise

* Aerospace Research Engineer
Vehicle Operations Research Branch

and high lift configuration are presented in figures 3 and 4, respectively. The data were collected at a dynamic pressure of 50 lb/ft² and a chord based Reynolds number of 3.3 million.

The scaling laws required to extrapolate the heavy rain wind tunnel data to full-scale have not been fully established. A series of full-scale airfoil tests in a simulated rain environment is being conducted at NASA Langley's Aircraft Landing Dynamics Facility (ALDF) in an effort to determine the scaling effects and to understand the primary influences and flow mechanics of this two phase flow environment.^{3,4} The ALDF is an outdoor test facility which is generally used for landing gear and tire studies but was modified for this series of tests. The facility consists of a large carriage which is propelled down a track. The full-scale airfoil section model is mounted atop the carriage. An array of spray nozzles is suspended above the track to simulate the rain environment. The carriage with the attached airfoil section is capable of speeds up to 170 knots. The rain spray system can produce water mass concentrations of 2, 9, 26, and 35 g/m³. A photograph of one of the heavy rain full-scale test runs is shown in figure 5.

Preliminary results of the ALDF full-scale tests show that the maximum lift coefficient and stall angle of attack are reduced in the same manner as the wind tunnel test results. A comparison of the wind tunnel and full-scale lift curves at similar rain rates is shown in figure 6. The agreement between the data sets may indicate a lack of significant scaling effects at these conditions. The ALDF tests are continuing at reduced rain rates to determine the sensitivity to rain rate and the minimum rate at which significant performance penalties exist.

Modeling Methodology

Currently, there is no experimental data base of the aerodynamic effect of heavy rain on a full configuration airplane. However, by assuming the scaling effects of the heavy rain wind tunnel data to be small, and integrating the airfoil section data across the planform of the airplane, an approximation of the full configuration effect can be developed.

The commercial twin-jet transport configuration selected for this study was that of NASA Langley's Transport Systems Research Vehicle (TSRV) shown in figure 7. An existing simulation model of the TSRV was used as the dry baseline model. The span-wise lift distribution of the airplane was used to weight the integration of the wind tunnel section data across the wing planform.

$$C_L = \frac{2}{b} \int_0^{\frac{b}{2}} S l(y) c l(y) dy$$

$$C_D = \frac{2}{b} \int_0^{\frac{b}{2}} S l(y) c d(y) dy$$

The airplane wing planform with the inboard and outboard flap locations are shown in figure 8, along with the span-load coefficients at various flap settings. The span-load coefficients were computed with a vortex-lattice computer code for the different flap positions.⁵

A comparison of the lift coefficient computed by the planform integration method for the dry condition with that of the baseline airplane model is shown in figure 9. As can be seen, the integration is not an accurate method of estimating the total lift of the airplane, particularly at the high lift flap settings. This can be attributed to a number of factors, two of which are: a) the wind tunnel airfoil model is not the same airfoil and flap configuration as on the airplane, and b) the vortex-lattice method used to compute the wing loading is not an accurate method for multi-element flap and slat configurations. Consequently, a direct modeling of the integrated results would not accurately represent the wet airplane aerodynamics. However, the wet airplane aerodynamics could be approximated by modeling the change in the lift and drag with liquid water content from the integrated results, and apply this perturbation model to the dry baseline aerodynamic model. This was the technique used in this study.

Two examples of the resultant wet airplane aerodynamic model are presented in figures 10 and 11 for a take-off and landing configuration, respectively. The decrease in maximum lift coefficient and stall angle of attack, which was prevalent in the airfoil data is also evident in the airplane model, but to a lesser extent than the high lift section data. There is also a considerable drag increase at the higher lift coefficients.

Heavy Rain Effect on Climb Performance

The performance impact of heavy rain was demonstrated by computing the airplane's climb performance under increasing rain concentrations. Figures 12 and 13 show the steady-state rate of climb as a function of airspeed with rain concentrations of 0, 10, 20 and 30 g/m³, for a take-off and landing configuration, respectively. The take-off and landing configurations were defined as follows:

	<u>Take-off</u>	<u>Landing</u>
Flaps	5°	25°
Gear	up	down
Gross Weight	100,000 lbs	89,000 lbs

The climb rates were computed at take-off thrust (~24,000 lbs) to simulate a wind shear escape condition.

Figures 12 and 13 show a loss in climb performance of at least 200 ft/min at rain concentrations of 20 g/m³ and greater, for both the take-off and landing configurations. The greatest performance loss occurred at the low speeds where the higher angles of attack were required.

The performance effect of heavy rain can be equated to that of wind shear through the wind shear "F-factor".⁶ The F-factor is a hazard index which represents the rate of specific energy loss due to wind shear and is defined as:

$$F = \frac{\dot{W}_x}{g} - \frac{W_h}{V}$$

Figure 14 shows the steady-state rate of climb as a function of airspeed at various F-factor values, for the take-off configuration. By cross-plotting figures 12 and 14, an equivalent F-factor curve can be derived for each rain concentration curve. Figures 15 and 16 show the equivalent F-factor as a function of airspeed for the take-off and landing configurations, respectively. For the 20 g/m³ case, the equivalent F-factor was about 0.02 for the majority of the speed range. An F-factor of this magnitude represents 20 percent of the currently accepted wind shear alert threshold. At the low speeds the F-factor was significantly greater. From figure 14, it can be seen that an F-factor of 0.16 or more may result in the airplane being unable to climb or maintain altitude. The 20 g/m³ case therefore represents a loss of at least 12 percent of the climb performance margin of the airplane.

In an effort to determine how much of the heavy rain performance degradation was due to the increased drag versus the decreased lift, the rate of climb was computed including only the rain effect on lift and excluding the effect on drag. Alternately, the climb rate was then computed including only effect on drag. The results of this analysis are shown in figures 17 and 18 for the take-off and landing configurations, respectively. The increased drag had the greatest effect on the climb performance, particularly at the higher speeds. At the lower speeds the loss of lift effect was more pronounced, but never accounted for more than half the total performance degradation.

Wet Wind Shear Recovery Performance

In the event of an inadvertent wind shear encounter, the FAA recommended wind shear recovery procedure may require the airplane to be operated at or near the stick shaker angle of attack.⁷ The results of the heavy rain airfoil tests showed that the rain can reduce the stall angle of attack. This generated some concern that in a wet wind shear encounter, the airplane may stall prior to the stick shaker angle of attack, requiring significant modification of

the recommended recovery procedure. Figure 19 shows the angle of attack margin between stick-shaker and stall at rain concentrations of 0, 10, 20 and 30 g/m³ for each flap setting. The stall margin may be reduced by as much as 50 percent at the highest rain concentration, but the airplane does not stall prior to stick shaker angle of attack.

To illustrate the effect of heavy rain on wind shear recovery performance a batch simulation of a wet microburst encounter was conducted. The point mass airplane simulation model of reference 8 was modified to include the microburst model of reference 9 and the heavy rain aerodynamic model described earlier. The simulation was conducted for a take-off and an approach to landing situation. The take-off case was initiated at an airspeed of 138 knots, an altitude of 10 feet, and the previously defined take-off configuration. The landing case was initiated at 300 feet, 137 knots, on a 3 degree glide slope in the landing configuration.

The axisymmetric microburst model had a maximum outflow of 37 knots at an altitude of 120 feet and a radius of 2,391 feet. The severity of the shear is representative of microbursts which have caused aircraft accidents. The rain was simulated as a step input when within the 2,391 foot microburst radius, at concentrations of 0, 10, 20 and 30 g/m³. The center of the microburst was located 3,000 and 4,000 feet down range of the starting point for the take-off and landing cases, respectively.

The wind shear recovery procedure used in the simulation, modeled the FAA recommended procedure. The recovery was initiated when the wind shear hazard index (F-factor) reached the alert threshold value (0.12). At that point the throttles were advanced to take-off thrust and the airplane pitched to an initial target attitude of 15 degrees. If the airplane was unable to avoid descending at the initial target attitude, the pitch was increased until level flight could be maintained. Throughout the recovery the target pitch was limited to the value corresponding to the airplane's stick shaker angle of attack. The pitch rate of the airplane was limited to 3 degrees per second.

The results of the simulated wet microburst encounter are presented in figures 20 and 21 for the take-off and landing cases, respectively. The higher rain concentrations had a considerable impact on the recovery performance of the airplane. At 30 g/m³, the airplane was unable to recover from an otherwise recoverable microburst encounter.

The sensitivity of the recovery performance to the increased drag versus the decreased lift was studied in a similar manner as was done in the climb performance analysis. Figures 22 and 23 show the sensitivity results at 20 g/m³ of rain for the take-off and landing cases, respectively. Again, the increased drag had the greatest effect on the overall recovery performance, particularly for

the take-off case. This is primarily due to the increased drag reducing the airspeed, thus decreasing the energy of the airplane and extending the exposure time in the wind shear.

Concluding Remarks

The results from earlier sub-scale and full-scale airfoil tests have documented a reduction in the maximum lift capability and stall angle of attack and a corresponding increase in drag with increasing rain concentrations. Based on the data from these earlier tests, a estimate of the heavy rain effect on a commercial twin-jet has been developed. The results of this analysis indicate that the aerodynamic penalty associated with heavy rain can substantially reduce the airplane's performance. A loss in climb performance of at least 200 ft/min was noted at rain concentrations of 20 g/m³ and greater. This represented a loss of at least 12 percent of the climb performance margin of the airplane. A performance loss of this degree can critically impair an airplane's ability to escape a performance limiting wind shear. The drag increase associated with the heavy rain had the greatest impact on the wind shear recovery performance. This was primarily due to the drag reducing the airspeed and thus extending the wind shear exposure time and reducing the airplane's energy state.

The analysis presented here was based on a very limited data set with some rather broad assumptions. This paper illustrates the need for further heavy rain testing of sub-scale and full-scale airfoil sections to determine the scaling relationships and flow mechanics involved. It also illustrates the need for full configuration testing. The effect of heavy rain on the fuselage and empennage was not included in this analysis and is expected to further degrade the airplane's performance.

References

1. Luers, J. K.; Haines, P.: *The Effect of Heavy Rain on Wind Shear Attributed Accidents*. AIAA-81-0390, January 1981.
2. Bezos, G. M.; Dunham, R. E., Jr.; Gentry, G. L., Jr.; Melson, W. E., Jr.: *Wind Tunnel Test Results of Heavy Rain Effects on Airfoil Performance*. AIAA-87-0260, January 1987.
3. Bezos, G. M.; Campbell, B. A.; Melson, W. E., Jr.: *The Development of a Capability for Aerodynamic Testing of Large-Scale Wing Sections in a Simulated Natural Rain Environment*. AIAA-89-0762, January 1989.

4. Bezos, G. M.; Dunham, R. E., Jr.; Campbell, B. A.; Melson, W. E., Jr.: *Results of Aerodynamic Testing of Large-Scale Wing Section in a Natural Rain Environment*. AIAA-90-0486, January 1990.
5. Margason, Richard J.; Lamar, John E.: *Vortex-Lattice Fortran Program For Estimating Subsonic Aerodynamic Characteristics of Complex Planforms*. NASA TN D-6142, February 1971.
6. Bowles, R. L.; Targ, R.: *Wind Shear Detection and Avoidance: Airborne Systems Perspective*. International Congress of Aeronautical Sciences, Jerusalem, Israel, September 1988.
7. Boeing Company: *Wind Shear Training Aid. Volume I - Overview Pilot Guide, Training Program*. Contract DFTAO-1-86-C-00005, February 1987. (Available from NTIS as PB88 127 196)
8. Hinton, David A.: *Flight-Management Strategies for Escape From Microburst Encounters*. NASA TM-4057, August 1988.
9. Oseguera, Rosa M.; Bowles, Roland L.: *A simple, Analytical 3-Dimensional Downburst Model Based on Boundary Layer Stagnation Flow*, NASA TM-100632, July 1988.

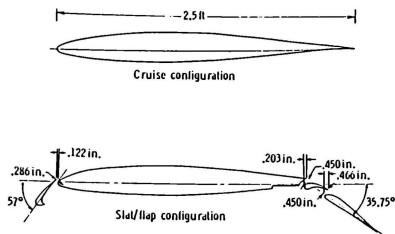


Fig. 1. Cross section of NACA 64-210 airfoil model and details of slat and flap installation.

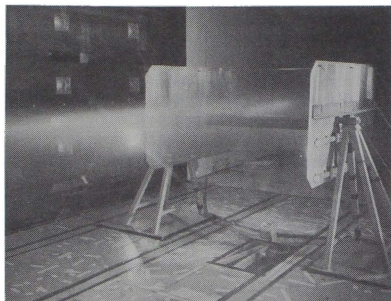


Fig. 2. Photograph of heavy rain airfoil test in NASA Langley 14- by 22-Foot Subsonic Tunnel.

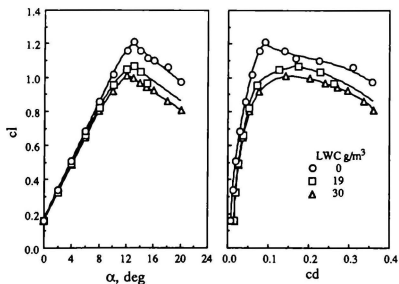


Fig. 3. Wind tunnel lift and drag measurements for the cruise configuration.

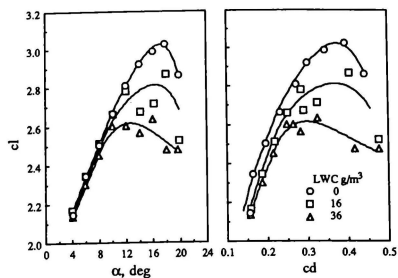


Fig. 4. Wind tunnel lift and drag measurements for the high lift configuration.

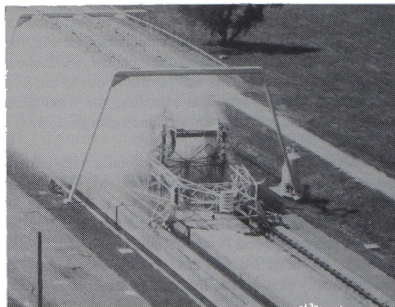


Fig. 5. Photograph of heavy rain full-scale airfoil test at NASA Langley's ALDF.

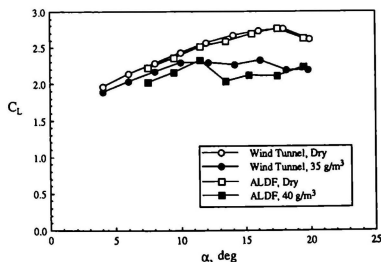


Fig. 6. Comparison of lift coefficient data from wind tunnel and ALDF tests.

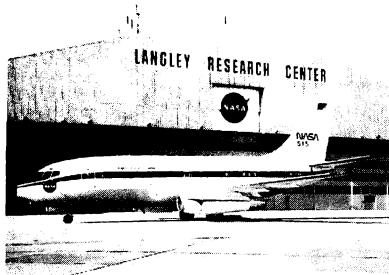


Fig. 7 Photograph of TSRV.

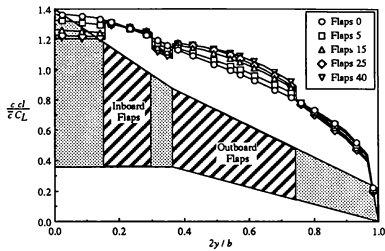


Fig. 8 TSRV wing planform and span-loading.

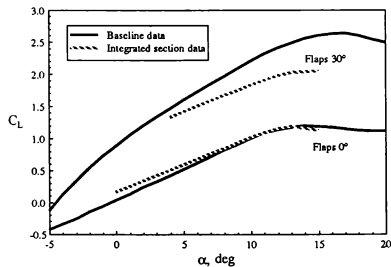


Fig. 9 Comparison of baseline and computed lift curve.

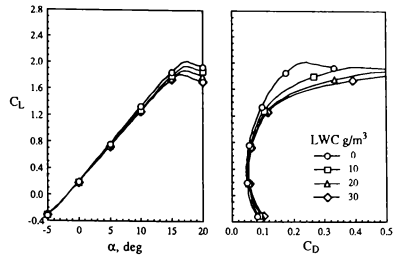


Fig. 10 Heavy rain aerodynamic model for airplane in a take-off configuration. (Flaps 5°, Gear up)

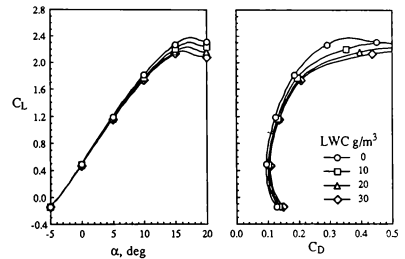


Fig. 11 Heavy rain aerodynamic model for airplane in landing configuration. (Flaps 25°, Gear down)

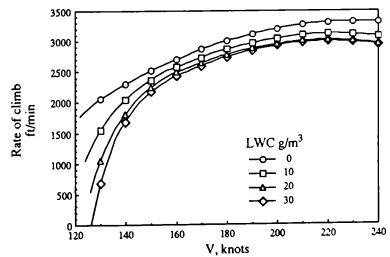


Fig. 12 Effect of heavy rain on climb performance in a take-off configuration.

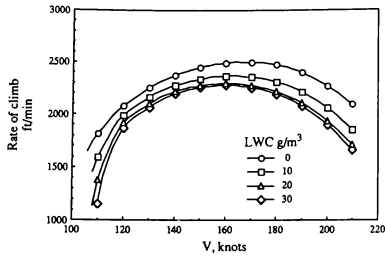


Fig. 13 Effect of heavy rain on climb performance in a landing configuration.

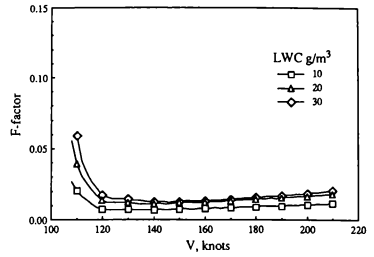


Fig. 16 Heavy rain equivalent F-factor in a landing configuration.

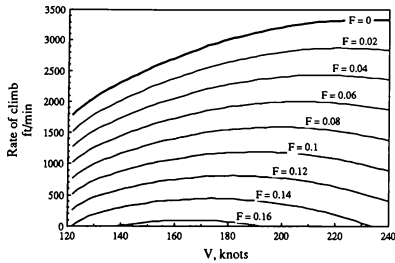


Fig. 14 Effect of wind shear on climb performance in a take-off configuration.

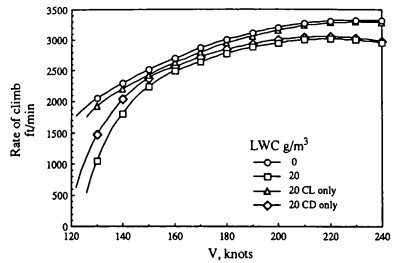


Fig. 17 Sensitivity of lift and drag on climb performance in a take-off configuration.

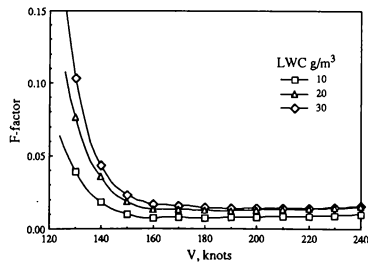


Fig. 15 Heavy rain equivalent F-factor in a take-off configuration.

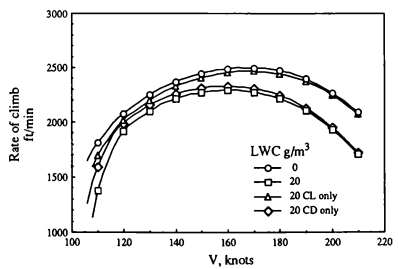


Fig. 18 Sensitivity of lift and drag on climb performance in a landing configuration.

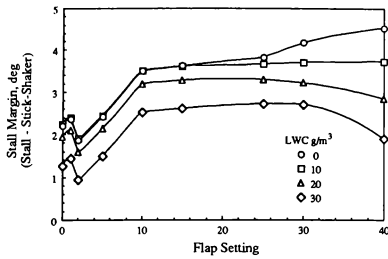


Fig. 19 Effect of heavy rain on stall margin.

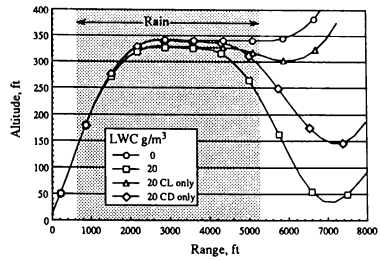


Fig. 22 Sensitivity of lift and drag on wind shear escape performance in a take-off configuration.

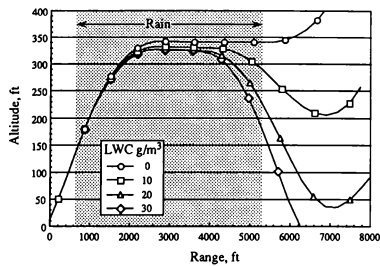


Fig. 20 Effect of heavy rain on wind shear escape performance in a take-off configuration.

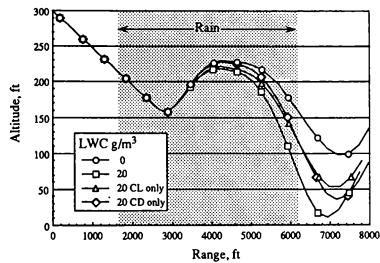


Fig. 23 Sensitivity of lift and drag on wind shear escape performance in a landing configuration.

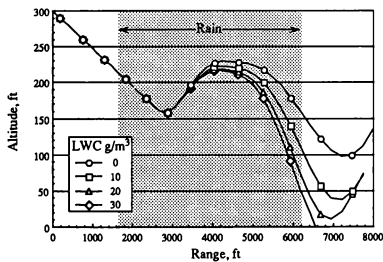


Fig. 21 Effect of heavy rain on wind shear escape performance in a landing configuration.

ADVERSE WEATHER SIMULATION CONCEPTS FOR SAFETY OF FLIGHT TRAINING

Bruce Montag
 Southwest Research Institute
 San Antonio, Texas

Abstract

Much attention has been focused on the problem of air travel safety and the hazard to aviation posed by flight through adverse weather. In response to the public's concern over these issues, the FAA has mandated that all commercial jet aircraft must have advanced windshear detection and flight escape guidance systems installed by January 4, 1993.⁶⁴ The purpose of these new avionics systems is to provide pilots with better information about the weather environment so that they may avoid encountering hazardous flight conditions that pose a risk to air travel safety.

Commercial airline pilots are trained in flight safety techniques and are evaluated against established standards of performance in commercial flight simulators. The current state-of-the-art in commercial aircraft simulation technology does not yet fully support all of the weather characteristics that are apparent to aircrews in flight. Current weather simulation technology focuses on weather radar display generation and airframe motion cues, with a loose coupling between the two simulations. Out-the-window visual display characteristics of dynamic weather patterns, one of the most important piloting cues, and integrated aerodynamic effects are key weather environment features that are currently not simulated for pilot training.

The advent of new weather processing avionics systems calls for a new look at the real time simulation of weather phenomenon for pilot training. Simulation of emerging airborne weather detection technologies such as scanning lidar, high resolution doppler radar, and forward looking infrared systems will not only involve cockpit display generation, but the development of a meteorologically correct, integrated weather model that is capable of supporting all weather characteristics that are apparent to the aircrew.

This paper provides an overview of weather sensor systems, flight training, current weather simulation approaches, adverse weather training and simulation requirements, and a new concept for a modular, adverse weather simulation system. This weather simulation concept provides a multi-sensor, correlated weather environment for advanced safety of flight training applications. The concept provides a means for controlling and correlating out the window visual weather scenes, weather processing sensor avionics displays, and aircraft handling qualities. A total weather environment simulation will allow adverse weather crew training to move from the classroom to the cockpit for greater effectiveness. The adverse weather simulation concept is based on the integration and synthesis of existing weather modeling research with rapidly advancing sensor simulation technology. The proposed weather concept provides a standard for weather simulation and offers a consistent means of presenting weather cues to the aircrew.

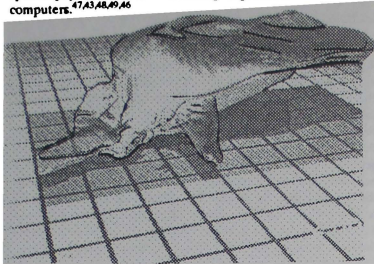
Nomenclature

N	Number of pulses per beam width
PRF	Pulse Repetition Frequency
P_r	Received signal power
P_t	Transmitted signal power
G	Radar gain
λ	Radar Wavelength
R	Range
L	Radar loss factor
σ	Radar reflectivity coefficient
τ	Radar pulse width
c	Speed of light
V_a	Maximum unambiguous doppler velocity
r_a	Maximum unambiguous doppler range
$E(t)$	Volt/meter electric field
$H(t)$	Amps/meter magnetic field
M	Dipole moment
ϵ	Capacitance ratio
d	Laser telescope diameter
β	Laser backscatter cross section
η	Laser detection and mixing efficiency
K(R)	Laser extinction for range R
B	Laser detector bandwidth
h	Planck's constant
F_s	Windshear hazard index
W_h	Horizontal wind speed rate
W_v	Vertical wind speed
g	Gravitational acceleration
V	Aircraft true airspeed
FOV	Field of View

Introduction

Weather simulation is beginning to play a larger role not only in flight crew training but also in the development of aircraft sensor systems. Under the Integrated FAA Windshear Program, simulation is used extensively for modeling and testing of candidate airborne weather sensor systems.³⁴ NASA-Langley has sponsored the development of a Terminal Area Simulation System (TASS) for the FAA program that characterizes adverse weather phenomenon such as microbursts. The TASS produces four dimensional data sets that encode weather parameters at discrete grid points distributed throughout a local volume of airspace.⁵⁰ The TASS model, discussed in further detail later, represents the state-of-the-art in weather modeling.

Meteorological models that simulate the dynamic behavior and measurable characteristics of storm clouds are becoming more available today due to a better understanding of atmospheric physics and the increasing capabilities of computers.^{47,43,48,49,46}



Wilhelmson's Numerical Cloud Model

The technology of weather simulation for flight training is still in its infancy. Currently, only rudimentary weather capabilities are available for real-time simulation. Advancing the state-of-the-art in weather simulation technology offers the payoff of increasing flight safety by transitioning weather related training from the classroom to the cockpit. By synthesizing 4D atmospheric science modeling with advanced visual and sensor simulation technology, a new concept for total, correlated weather simulation emerges.

Current airborne weather sensors include radar, lighting detectors, laser radar, forward looking infrared, the airframe, flight instruments, and human vision. Effective employment of these sensors by aircrews to avoid flight into adverse weather requires training. An overview of these systems and an analysis of current training practices and weather simulation systems are presented below followed by a discussion on adverse weather training and simulation requirements, and a concept description of an integrated weather simulation approach.

Weather Sensing Airborne Systems

In 1983, the National Research Council Committee on Low Altitude Wind Shear and Its Hazard to Aviation recommended that forward looking airborne sensors be developed to allow for the remote detection and avoidance of windshear along an aircraft's flight path.²⁸ In response to this call, the FAA initiated the Integrated FAA Windshear Program Plan that includes the assessment of candidate forward looking airborne sensors that are capable of detecting wind shear along the flight path.³⁶ A variety of sensors are being evaluated to meet this objective. The key sensor feature needed is the ability to infer the spatial distribution of wind velocity components in the airspace ahead of the aircraft. Weather information other than wind parameters is also very valuable to the aircrew. Knowledge of rainfall rate, cloud top location, weather movement direction, and stage of storm development are all important for in-route flight planning. The following sensors all convey key informa-

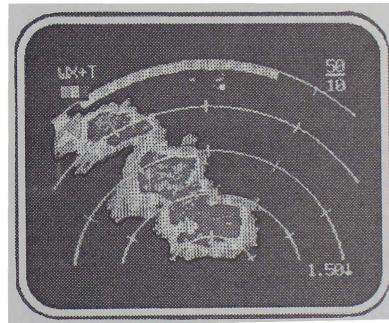
tion to the aircrew, and an understanding of how they sense the weather environment is essential for the development of an integrated weather simulation model.

Radar

Large, magnetron type pulsed C-Band radars were first applied on a few commercial airliners in the 1950's to detect rainfall ahead of the aircraft.⁸ Today, small low power solid state weather radars are commonplace and are available for virtually any aircraft type. In the 1980's, airborne turbulence detecting radars were developed that utilize doppler processing of reflectance data to provide a coarse assessment of wind speed variability within the scanned volume of airspace along the flight path.³⁴ Both radar types, pulsed and doppler, drive cockpit visual displays that indicate the sensed weather situation to the aircrew.

Pulsed Radar

Most weather radars in operation are of the incoherent, pulsed transmission variety. The cockpit display for these radars typically provides a color coded plan-position sector view between a set of azimuth scan limits for a given antenna elevation and range scale selection. Each display pixel within the azimuth scan area represents a radar range bin. The color assigned to each pixel corresponds to the sensed radar reflectivity value for that range bin.



Weather Radar Display

The reflectivity displayed is proportional to the number and size of rain drops (i.e. the liquid water content) appearing within the pulse resolution volume of air defined by the radars beam width and pulse length. The number of pulses per beam width area is given by equation 1.

$$N = \frac{PRF \times Beamwidth}{Scan Rate} \quad (1)$$

The radar processor integrates the number of pulse returns within the beam width for each range bin to yield an accumulat-

ed average reflectance for each bin.²⁵ The range bin reflectance value is a function of the return pulse signal power received by the radar. Equation 2 is the standard weather radar range equation that shows the relationship between radar parameters and weather characteristics.

$$P_r = \left(\frac{P_t G^2 \lambda^2}{(4\pi)^3 R^4 L} \right) \sigma \quad (2)$$

Sigma (σ) is the backscatter coefficient sum of all rain drops within the pulse resolution volume. It is a function of rain drop size, radar wavelength, refraction index, and rainfall rate. The range resolution of each display pixel is a function of the radar pulse width, the pulse repetition frequency, and the radar screen display resolution, as indicated in equation 3.

$$\text{Max. No. of Range Bins per pulse} = \frac{1}{\text{PRF} \times \tau} \quad (3)$$

The relationship between range bins and display pixels is a function of the number of pixels available for display. When the number of range bins exceeds the display resolution, the range bin reflectance data is averaged to yield an aggregate pixel.²⁵ Some radars also provide either an expanded sector option or an adjustable pulse width that allows a one-to-one correlation between range bins and pixels. The range resolution of each range bin is roughly 500 feet for every microsecond of pulse width, as indicated in equation 4.

$$\text{Range resolution} = \frac{c \tau}{2} = 500 \text{ ft per } \mu\text{sec} \quad (4)$$

Most color weather radars today feature pulse widths of less than 10 microseconds, with range resolution of better than 5000 feet per range bin.

Doppler Radar

In addition to providing the aircrew with the situational information concerning precipitation that pulsed radar provides, doppler radars provide information on rain pattern wind velocity towards and away from the radar and velocity dispersion (turbulence).⁴ Doppler radars sense relative velocity by measuring the doppler phase shift between two received radar pulses from the same range bin.²⁴ This technique is known as pulse pair processing. Doppler radars are coherent since the phase of the solid state radar transmitted signal is coherent from one pulse to the next.³⁰ By looking at phase shifts, a doppler radar can measure reflectance target velocities toward or away from the radar for each range bin. By comparing velocities between range bins (performed by some ground based radars), the large scale wind variation or shear within the radar scan volume can be determined and indicated to radar operator.⁵⁰ Doppler radars are limited by the maximum velocity spectral width and maximum unambiguous velocity that they are able to detect as a function of PRF as defined in equation 5.

$$v_a = \frac{\lambda}{4 \times \text{PRF}} \quad ; \quad v_a r_a = \frac{\lambda c}{8} \quad (5)$$

Current airborne doppler radars can measure wind borne rain drop velocity dispersion, but they do not attempt to measure wind vectors.⁶⁰ Velocity dispersion greater than 5

meters per second is the established indication of turbulence. Doppler capable radars such as the Collins TWR-850 and the Bendix RDR-4A use magenta colored pixels to indicate when turbulence within a range bin is detected.

Lightning Detectors

Another class of weather sensor is the thunderstorm or lightning detector such as the 3M Stormscope Series II. Lightning discharges are caused by the accumulation of negatively charged particles in the base of convective thunderstorms.⁵⁶ Inside a thundercloud, light positively charged particles (P region) are drawn upwards by wind currents while heavy negatively charged particles (N region) are drawn downwards by gravity. Lightning occurs when electrical discharge occurs between the N region and the ground.⁵⁶ Typically, intra-cloud lightning occurs five to ten times more often than cloud to ground lightning.⁵⁶ Frequent lighting is indicative of a mature convective thundercloud that may possess turbulent winds and may produce heavy precipitation.⁵²

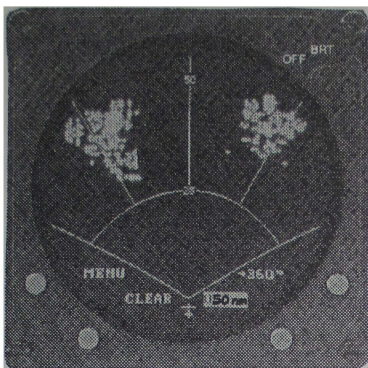
The Stormscope WX-1000 model operates by detecting electromagnetic and or electrostatic signals emanating from atmospheric electrical discharges, and displays them by range and bearing on a CRT display.⁶¹ Ranging to within 10% of range scale and bearing to within ± 5 degrees is approximated by analyzing the amplitude of the received electromagnetic and electrostatic discharge waveforms.⁶² Distance to the discharge event is determined by monitoring the time domain response of equations 6 and 7.⁶¹

$$E(t) = \frac{1}{4\pi\epsilon} \left(\frac{M}{R^3} + \frac{dM/dt}{cR^2} + \frac{d^2M/dt^2}{c^2R} \right) \quad (6)$$

$$H(t) = \frac{1}{4\pi\epsilon} \left(\frac{dM/dt}{cR^2} + \frac{d^2M/dt^2}{c^2R} \right) \quad (7)$$

Display Characteristics

The WX-1000 can indicate electrical discharge activity in either a 120 or 360 degree field of view out to 200 nm away. Up to 512 discharge events can be displayed, with 256 display locations reserved for the forward 120 degree sector.⁶² A plus sign is generated at the range and azimuth display location corresponding to the sensed discharge point, and the symbol will be deleted from the display after two minutes unless more electrical discharge activity occurs at that location.⁶² An available heading stabilization feature preserves the displayed weather map orientation during aircraft maneuvers and provides for increased weather situation awareness.



Stormscope WX-1000 Display

Lidar

Lasers can also be used in a fashion similar to doppler radar to detect the direction and speed of liquid water within clouds that is moved by the wind. In airborne doppler Lidar systems, a coherent, optical pulse train is transmitted through a gimbaled telescope that is scanned ahead of the aircraft. The laser pulses are scattered once they reach cloud cells and they are attenuated by the round trip through the atmosphere. The scattered pulses are received through the telescope and are concentrated on a photo-detector. The frequency shift of the detected signal is then analyzed to yield a doppler velocity.³⁴

Laser ranging capability is a function of the backscatter cross section, the transmitted pulse energy, the diameter of the telescope, and atmospheric conditions. The laser ranging equation is given below.³⁴

$$P_r = \frac{P_t \pi d^2 \beta \lambda \eta K(R)}{\delta R^2 B h} \quad (8)$$

Lidar differs from radar in that it is capable of precise velocity and range resolution, but is also range limited (< 3km) and is severely affected by atmospheric conditions.³⁷

FLIR Radiometers

Another promising adverse weather sensor is the forward looking infrared (FLIR) radiometer. The FLIR approach is totally passive since atmospheric emissions in the IR region are received and processed without the need for a transmitted and reflected signal pulse. Unlike radar and lidar sensors, FLIR radiometers can detect and measure wind speed independent of the presence of precipitation.³⁴

Windshear detecting FLIRs operate by remotely measuring the amount of energy emitted in the IR spectrum by a volume

of air within the FLIR field of view.¹⁵ The received energy is then translated into an equivalent mean temperature reading. The corresponding look distance for this temperature reading is a function of the IR wavelength. FLIR sensor range is selected through wavelength adjustment, similar to range gating a radar.³⁴ Spatial wind speed within the FLIR field of view is determined by calculating the temperature rate of change across a set of look distances (dT/dt).³⁴ It is conceivable that by stepping the FLIR through a range/azimuth scan, that a spatial wind map could be created for aircrew situational awareness. Current research has established that a windshear alert condition exists when the temperature rate of change exceeds 0.5 degrees per second over a period of 8 seconds.³⁴

FLIRs can also be used to detect other meteorological phenomenon that may be hazardous to flight such as clear air turbulence (CAT). CAT can be detected by sensing the water vapor that becomes concentrated into horizontal, isothermal layers due to vertical wind shear.¹⁵ Through wavelength and processing algorithm selection, a FLIR radiometer can therefore be configured for in-route hazard avoidance as well as terminal area windshear avoidance.

Drawbacks to the FLIR weather sensor are that little is currently known about IR atmospheric noise or about the probability of detection and false alarm performance that can be expected of an adverse weather sensing FLIR in operation.³⁴ In addition, atmospheric emissivity is heavily attenuated and can be masked completely by intervening heavy rain, although the FLIR range capability is still about 4 times greater than human vision.^{15,16}

Pilot Vision

The most important sensor for detecting adverse weather is the pilot's vision. Pilots can infer safety critical information about the surrounding weather environment by scanning out the window for key visual hazard indicators such as concentrated heavy rain, virga (rainfall that doesn't reach the ground), curling rain outflow, or diverging rain patterns.⁴⁰

Reading the Sky

The ability to read the sky is also important for in-route flight planning. Visible cloud characteristics such as the color, size, shape, and height of the cloud indicate the type of weather to be expected. For example, cumulonimbus clouds (heavy and dark, with a very high anvil shaped top) indicate potentially severe weather composed of heavy rain, high winds, and hail. Fractostratus clouds (light shaded, large coverage, layered clouds broken up into groups) at altitude indicate the presence of turbulent winds, high cirrus clouds (detached, fibrous, silky appearance) indicate icing conditions, and ground fog indicates still air with little chance of rain.⁵¹



Visible Cloud Characteristics

The pilots ability to visually resolve cloud characteristics is a function of sun position (i.e. light level and direction), the clarity of the intervening atmosphere, the size of the cloud, the range from the observer to the cloud, and the degree of contrast between the cloud and the sky.⁶¹ If you are staring at a given section of the sky, your binocular visual field of view is approximately 60 degrees. For a 1M square foot sized cloud that exhibits a cloud to sky contrast ratio of .2, the cloud will be visible within your field of view up to 5 miles away.⁶¹ Within the terminal area (under 3 miles), specific weather effects such as rain fall direction and wind effects such as blowing dust devils become more apparent.

Airframe

The aircraft airframe itself has proved to be a useful sensor of adverse weather conditions as well. Several reactive wind-shear warning systems are in operation today that sense horizontal and vertical wind speed and wind speed rate of change at the aircraft.³⁷ Wind speed rate is determined by comparing sensed airspeed rate and computed groundspeed acceleration. The hazard index defined in equation 9 has been developed to detect when the aircraft has encountered a hazardous windshear condition³⁷

$$F = \frac{\dot{W}_x}{g} - \frac{W_x}{V} \quad (9)$$

A threshold value of the hazard index F is determined for a particular aircraft's performance capability, though there is a fine line governing threshold sensitivity. Too low of a threshold will result in troublesome false alarms while too high of a threshold will result in an after the fact warning that will be too late for the aircrew to take appropriate action. The horizontal and vertical wind velocity components are sensed by comparing air mass acceleration with inertial aircraft acceleration. Hazardous windshear occurs when the time rate of change of wind speed varies rapidly increasing in absolute magnitude. This rapid change of the relative wind results in corresponding angle of attack and stick force changes that are independent of pilot

input.⁶⁰ In addition to pure wind vector changes, high frequency wind speed variability or turbulence may also buffet the airframe and complicate aircraft handling.

Flight through heavy rain also complicates airframe handling. Although results have been inconclusive to date, heavy rain is thought to have a measurable effect on airframe aerodynamics.¹⁸ In addition to the possibility of reducing lift capability, rain adds weight to the airframe (although only a marginal amount), induces momentum transfer effects, adversely affects engine performance, severely limits visibility, and increases cockpit confusion due to precipitation impact noise.⁶⁰

Instruments

Barometric altimeters will provide anomalous readings during flight through adverse weather due to rapid pressure variations. For instance, flight into a storm cell may result in the barometric altimeter indicating a climb due to decreasing atmospheric pressure, when in reality the aircraft is flying at a constant geometric altitude.

Vertical speed indicators (VSI) are also subject to error during adverse weather operations. VSI's operate by sensing the rate of change of static air pressure. Instrument lag is the most serious problem associated with VSI's, although inertial based VSI's are better than air data based instruments. Air data based instruments typically lag the aircraft by 4 seconds during windshear escape maneuvers, while inertial systems lag by around 2 seconds.⁶⁰

Indicated airspeed will also vary widely with atmospheric pressure fluctuations and apparent wind vector since the airspeed indicator measures dynamic pressure directly.

Adverse Weather Flight Training

Training requirements for the certification of commercial airline pilots is addressed in the Federal Airline Regulations (FAR) part 121. Amendment 121-199, which went into effect on January 2, 1989, requires that low altitude windshear flight training programs be incorporated into part 121 certificate holders approved training programs.⁶⁴ In 1987, the Windshear Training Aid, a set of instructional materials developed by a Boeing led industry team, was distributed by the FAA to facilitate the development of these adverse weather related flight training programs.

Windshear Training Aid

An important element of the Integrated FAA Windshear Program was the development of a training course to familiarize pilots with adverse weather and its effects on aircraft. Known as the Windshear Training Aid, this FAA approved courseware includes video tapes, classroom instructional materials, and adverse weather reference information. The Windshear Training Aid includes windshear substantiating data, a pilot windshear guide, an example generic windshear training program, and implementation guidelines for tailoring airline specific programs. The example training program is divided between a ground school segment and a simulator training segment. The ground school segment primarily addresses windshear avoidance and

provides training in the meteorology of adverse weather and discusses aircrew procedures.³⁹ The simulator segment primarily addresses windshear flight recovery techniques once windshear has been encountered.⁴⁰

The Windshear Training Aid includes a limited discussion of visual adverse weather indicators and weather radar usage, but does not go far enough. The name of the game in adverse weather training is avoidance. To be truly effective, an adverse weather training program needs to concentrate on developing aircrew detection and recognition skills to avoid flight into hazardous weather. Although training in recovery techniques may be the key to last minute survival, training in the use of emerging sensors and sensor techniques to avoid flight into adverse weather is just as critical for flight safety. This type of training will increase in importance as sensors and sensor capabilities mature and move into daily aircraft operations.

Weather Simulation Systems

The state-of-the-art in weather simulation for flight training is reflected in several different approaches to providing a simulated weather environment. In most flight simulators that have been developed over the last ten years, the simulation of weather effects is distributed across many system simulation models. Typically, a static set of weather conditions that vary with altitude is selected for the particular flight training session. Each simulation model (i.e. engine, airframe, radar, etc...) then senses the selected weather conditions and determines the effect of the weather on that systems performance. The drawback of this approach is that the weather conditions around the airframe (i.e. temperature, pressure, wind vector) are not time varying and may not be matched meteorologically unless the instructor is "weatherwise". Additional problems are presented in the simulation of forward looking sensors. Currently, most flight simulators offer only a very crude weather effects correlation between the out-the-window scene and forward looking sensor views for radar, infrared, and electro-optical devices. This deficiency exists today in part because the FAA's Advanced Simulation Plan does not currently require cue synchronization or meteorological correctness of simulated weather effects.⁶⁴

Thunderstorm Environment Model

The Thunderstorm Environment Model (TEM) is a weather simulation system developed by Singer-Link Flight Simulation Division to provide a more realistic representation of weather for line oriented flight training (LOFT).^{10,12} The TEM simulation is based on a weather characteristics database that can be flown through and sensed by the airframe.^{10,12}

The TEM weather database consists of a three dimensional gridded data set that is updated every three minutes in simulator operation.¹² The data set is derived from ground based radar observations of a storm system, and is stored in a JAWS type data format. The Joint Airport Weather Studies (JAWS) project was first performed by the National Center for Atmospheric Research (NCAR) in 1982 and recorded several storm events using multiple doppler radars.³⁰ The recorded radar velocity and reflectivity data was then transformed into a three dimensional cartesian data set.²² The TEM weather data set utilizes a .8

kilometer interval between data points along each axis, and covers a 20nm x 20nm x 3200 foot volume of airspace.¹² Tri-linear interpolation is used for data smoothing and a 2nd order turbulence closure model is used to model windspeed variability as the aircraft flies through the weather affected airspace.¹² A microburst data set is also overlayed within the storm data set to provide for controllable windshear recovery training. The microburst model is composed of 10 data files that are selected at three minute intervals.¹² Each grid point within the weather data set describes twelve weather parameters consisting of: three components of wind, three turbulence rms values, two turbulence scale lengths, temperature, pressure, precipitation rate, and meteorological visibility.¹²

Although a vast improvement over traditional weather simulation methods, the TEM simulation does not provide a total solution for adverse weather flight training. Correlation between the weather simulation and the simulator visual system is limited to visibility range and storm center location, which offers little towards weather recognition and avoidance training.¹² Correlation between the 4D atmospheric environment and the weather radar simulation is also tenuous since it must be performed offline, owing to the radar simulations unique data format and database structure (cells and slices). The inclusion of other sensor types into the simulation would also be difficult to accomplish in a correlated manner. The slow temporal update of the weather environment and large spatial data intervals may present sensor simulation problems, particularly with the visual system. Use of observed and recorded data sets may also present problems in terms of accomplishing specific weather phenomenon detection and avoidance training objectives that require specific meteorological features that may not appear in the recorded data.

Weather Radar Simulation

Current color weather radar simulations use simplified geometric models of rain clouds to generate the radar display. One approach utilizes a three dimensional cubic grid to represent crude cloud patterns.¹¹ A static weather radar database is created by "coloring in" cells on the grid to determine cloud span and height. The simulated radar beam position is then scanned across the grid and slant range converted to determine the radar reflectance for each radar range bin. This approach results in a very chunky looking radar display that is not meteorologically correct and offers limited training value.

Another approach to weather radar simulation employs reflectivity contour cells as the weather database.¹³ Typically, radar reflectivity contours are digitized from a plan view ground based radar map. Each set of contours forms an altitude slice view of the storm cloud. These altitude slices are then linked together to form a cloud cell. Cloud cells are then linked to form an integrated storm system. The cloud cells and altitude slices can be controlled in real time to exhibit the growth, decay, translation, and rotation characteristics of typical storms.¹³

The drawback to the contour database approach is that correlating the weather radar database with other simulator weather characteristics is a laborious and time consuming

process. It is also difficult to correlate training significant radar features between the radar simulation slice/cell format and the cartesian grid format of the atmosphere data.

Visual Weather Simulation

The simulation of meteorological based visual weather effects is almost non-existent in current visual system offerings. Most visual systems concentrate on terrain and target modeling with little attention given to atmospheric effects, other than those features called out in the FAA Advanced Simulation Plan. Almost all visual systems provide integrated or broken cloud decks, fog and haze, variable visibility range, and textured sky.^{57,58} Some visual systems such as the Evans & Sutherland SPX provide a generic thunderstorm cell with lightning flash that can be positioned by the host simulator within the visual database.⁵⁸

What is missing in the visual system area is the capability to correlate visual atmospheric effects with an integrated weather simulation. Since most adverse weather training programs focus on windshear recovery technique (which is primarily an IFR maneuver) this lack of correlation has not been a problem. As training programs transition to weather recognition and avoidance training, visual correlation will become a training necessity.

Terminal Area Simulation System

The Terminal Area Simulation System (TASS) is a complex, offline computer model of the atmospheric physics involved in windshear producing microbursts. TASS was developed in 1987 to aid the FAA Integrated Windshear Program in characterizing adverse weather phenomenon, and to provide realistic data for real-time flight simulations.^{50,65} The TASS model is designed to generate three dimensional data sets similar to the JAWS data, except on a much smaller grid scale of 40 meters.⁵⁰ The TASS model is initiated by constructing a set of initial atmospheric conditions at the models' top boundary. Example initial conditions include the radius of precipitation, the type of water content (rain, snow, hail) and the precipitation mixing ratio.³⁷ The model then calculates the microphysical interactions of the air mass over the gridded computation domain as a function of time using compressible, newtonian fluid dynamics equations.^{37,50,65} Any type of cumulonimbus weather behavior including microbursts, hail, and heavy rain can be initiated through selection of the boundary conditions.³⁷ The TASS model computes 11 parameters for each grid point, consisting of: three wind velocity components, pressure, temperature, liquid water content, ice content, precipitation rate, snowfall, and hail fall. The purpose of the TASS model is to increase the spatial resolution of the adverse weather wind field for flight simulation purposes and to create new databases to augment the JAWS derived data.³⁰

The TASS simulation is implemented in FORTRAN and is vectorized for use on the NASA-Langley VPS-32 supercomputer. For a 63 x 63 x 32 element grid, the ratio of simulation time, which is the time scale of the model, to computation time on the VPS-32 is 2:3 (i.e. slower than real time).⁶⁵ For a

43 x 43 x 27 element grid, the ratio is 2:1 (i.e faster than real time).⁶⁵

Adverse Weather Training Requirements

Training aircrews to recognize and avoid adverse weather phenomenon calls for an integrated weather effects simulation that provides synchronized, correlated weather cues to the aircrew. Simulated sensor indications, instrument readings, airframe and flight control response, and the out-the-window scene should all be consistent with one another in sensing the weather environment. All weather effects that can be expected in flight (i.e clear air turbulence, icing, hail, the jetstream) for all flight regimes in addition to the terminal area should be accommodated by the weather simulation model. An effective weather simulation model must be able to provide the required cues for training, and must be capable of correlating these cues across all simulated systems.

Visual Weather Cues

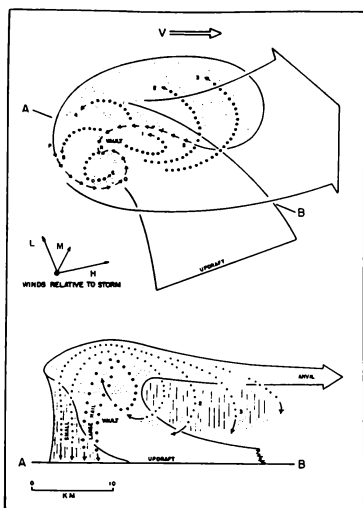
The Windshear Training Aid has pointed out that many of the key indications of severe weather are visual signs. It is important to note that most of these cues are not provided by current visual system offerings. Windshear related visual signs include blowing dust devils (indicates vortex winds), visible lightning emanating from cumulus clouds (indicates possible severe turbulence), rain shafts that do not reach the ground (indicates virga and a strong wind field), curling rain outflow near the ground (indicates a microburst), and horizontal range divergence from a vertical rainfall core (indicates horizontal windshear).⁶⁰



Microburst

Weather Radar Cues

A skilled operator can use a color weather radar to successfully detect many phenomenon that are key indicators of severe weather. Through interactive training, the radar operator can learn to interpret the development stage of storm cells, sense storm speed and direction, detect the probable presence of hail, detect radar hooks and vaults that indicate areas of low level air inflow and vertical updrafts, and detect shadows or attenuation caused blind spots that contain unknown weather.^{51,3} For total training, the model that drives the simulated weather display needs to be highly correlated with other system simulations and tightly integrated with the adverse weather simulation.



Radar Significant Features

Cue Synchronization

All weather related cues that are apparent to the aircrew should be temporally synchronized and spatially correlated to provide a complete weather training environment. The lack of this integrated capability is one of the primary reasons that adverse weather detection, recognition, and avoidance training is currently carried out in a classroom setting rather than in the simulator cockpit. For example, an effective adverse weather capable simulator would allow a pilot to scan a volume of sky with his radar, locate and steer towards a storm cell, acquire the same storm cell visually on the horizon, compare the visual and RF signature of the cloud, penetrate the cloud, and experience realistic wind speeds, turbulence, atmospheric variations, and precipitation effects on the airframe and instruments as the storm cell is transited. The benefit of this level of weather fidelity is that aircrews could be instructed in adverse weather detection and avoidance techniques, as well as windshear recovery methods.

Weather Environment Simulation Concept

To meet the need for an integrated weather simulation model, a new weather simulation concept emerges that synthesizes the current understanding of atmospheric modeling with advanced sensor simulation techniques. A sophisticated meteorological model such as TASS could be used to generate a high spatial and temporal fidelity weather database to support a given

training mission. In simulator operation, a Weather Effects Correlator (WEC) would manage and control the distribution of weather data to the various sensor simulations (i.e. visual, radar, airframe, etc.) in real-time, and thus ensure dynamic weather correlation for consistent cueing to the aircrew.

Weather Model Structure

Rather than distributing the simulation of weather effects among the various simulator components such as the visual system, the radar subsystem, and other subsystems, the integrated simulation approach is based on a centralized weather database that maintains all detectable characteristics of the weather environment. These characteristics are distributed to sensor simulations on an as needed basis, so that only the airspace within the instantaneous sensor field of view or in the vicinity of the airframe is considered. Weather characteristics for a given volume of airspace are computed ahead of the sensor line of sight, so that the 3D weather characteristics are already computed and are ready for transfer to the sensor simulation for inclusion in sensor processing.

A centralized weather simulation guarantees weather effects correlation, similarly to way that the generic transformed terrain database (GTDB) of Project 2851 is geared to ensure terrain correlation among sensor systems. Although targeted for weather sensor simulation, this weather environment simulation concept could also be applied to mission rehearsal applications that cover a broader range of sensors.

Database Resolution

In order to achieve correlation, the database resolution needs to match the detection resolution of each simulated sensor. The main idea is that sensor correlation, as accomplished by the WEC, is automated and performed online during the simulation. This approach standardizes the way that weather is sensed, and allows new weather databases to be incorporated for training without changing simulator software such as the radar database or the visual database.

Sensor correlation is accomplished by having each sensor simulation access the weather database as the data is needed. The key to this approach is that the weather data must be provided at the same resolution as that of the modeled sensor. For a weather radar simulation, the data needs to be distributed at range bin resolution (< 5000 ft grid spacing). For visual systems, the resolution requirement is much more demanding. Distant weather features (over 10 miles away) can easily be generated on an FAA Phase II visual (75 degree FOV @ 1280h pixels) from a 20 meter spaced data set since that spacing matches the span for each display pixel at that range.

$$\text{pixel span} = \text{range} \times \left(\frac{\text{FOV}}{\text{pixel resolution}} \right) \quad (10)$$

A much more difficult problem occurs at the shorter visual ranges encountered in the terminal area, where visual detection and recognition of weather effects is most important.

An approach to solving the resolution and correlation issue is to transform the weather data set within the sensors field of view to the sensors format and resolution needs. The Weather Effects Correlator would perform this function in real-time for the given sensor. Each sensor would receive weather data from the WEC in the format (i.e. scan line, spherical, cartesian) required for pipeline processing. To accommodate this integrated weather simulation approach, weather radar simulators and visual systems would need to include the weather data stream as another database input into the display generation pipeline.

Conclusion

With enhanced spatial and temporal weather data resolution and a standardized sensor correlation technique, 4D numerical atmospheric physics models offer much promise for enhancing adverse weather training, especially in the areas of weather detection, recognition, and avoidance. As computer processing capabilities expand, and the modeling characterization of the atmosphere improves, more capable weather simulations will be developed that can enhance our understanding of weather phenomenon and facilitate training in how to deal with weather situations in the flight environment. The integrated approach to sensor correlation presented here is a step towards making this capability a reality.

References

- 1) "Line Wind Shear Generation for Flight Simulator Applications." A.B. Markov. Journal of Aircraft. July, 1982.
- 2) "NASA Will Study Heavy Rain Effects on Wing Aerodynamics". Edward Phillips. Aviation Week & Space Technology. February 13, 1989.
- 3) "Use, Nonuse, and Abuse of Weather Radar". Edwin Kessler. Journal of Aircraft. May, 1988.
- 4) "Thunderstorm Generated Solitary Waves: A Wind Shear Hazard". R.J. Doviak, Journal of Aircraft. May, 1989.
- 5) "Modeling Clear-Air Turbulence with Vortices Using Parameter-Identification Techniques". Rajiv Mehta. Journal of Guidance. January-February, 1987.
- 6) "Overview of the Integrated FAA Wind Shear Program Plan". George Hay. Society of Automotive Engineers. Paper No. 861702.
- 7) "Pulsed Doppler Radar Detects Weather Hazards to Aviation". D.S. Zmic. Journal of Aircraft. February, 1982.
- 8) "New Airborne Weather Radar Systems". G.A. Lucchi. Journal of Aircraft. March, 1982.
- 9) "Meteorological Inputs to Flight Simulators". John Klehr. Journal of Aircraft. January, 1983.
- 10) "Simulator Recreates Realistic Weather Radar". Kenneth Stein. Aviation Week and Space Technology. January 31, 1983.
- 11) "Low Cost Weather Radar Simulation". Dennis Cowdrey. 5th Interservice Trainer Equipment Conference. November, 1983.
- 12) "A Four-Dimensional Thunderstorm Model for Flight Simulators". John Klehr. 5th Interservice Trainer Equipment Conference. November, 1983.
- 13) "Simulating Growing Thunderstorm Echoes for Weather Radar Training". John Klehr. 7th Interservice Trainer Equipment Conference. November, 1985.
- 14) "FAA Moves Out on Solving Windshear Problem". Interavia. February, 1989.
- 15) "Airborne Infrared System Provides Advance Warning of Turbulence". Aviation Week and Space Technology. June 19, 1989.
- 16) "Weather Performance Projection for RF, IR, and EO Systems". Fred Wilcox. IEEE International Radar Conference. 1980.
- 17) "Commercial Airborne Weather Radar Technology". G.A. Lucchi. IEEE International Radar Conference. 1980.
- 18) "NASA Tests Indicate Heavy Rainfall Can Reduce Lift at High Angles of Attack". Edward Phillips. Aviation Week and Space Technology. August 28, 1989.
- 19) "Researchers Developing Airborne Flir with Ability to Pinpoint Microbursts". William B. Scott. Aviation Week and Space Technology. February 19, 1990.
- 20) "Effect of Spatial Wind Gradients on Airplane Aerodynamics". Dan Vicroy. Journal of Aircraft. June, 1989.
- 21) "A Ring-Vortex Downburst Model for Flight Simulations". Journal of Aircraft. Michael Ivan. March, 1986.
- 22) "Aircraft Performance in a JAWS Microburst". Walter Frost and Ho-Pen Chang. Journal of Aircraft. 22, 1985, pp. 561-567.
- 23) "Model of the Wind Field in a Downburst". Shangxiang Zhu and Bernard Etkin, Journal of Aircraft. 22, 1985, pp. 595-601.

- 24) "Detecting Turbulence with Weather Radar", Eugene Schwarting, Avionics, November, 1989.
- 25) "Modern Colour Display and Processing System for Meteorological Radars", N.A. Cunningham, IEE Proceedings Vol. 128, Pt. F, No. 1, February, 1981.
- 26) "Thunderstorm Turbulence and its Relationship to Weather Radar Echoes", J. Burnham and J.T. Lee, Journal of Aircraft, Vol. 6, No. 5, Sept.-Oct., 1969.
- 27) "Next Generation Weather Radar", Microwave Journal, January, 1990.
- 28) Low-Altitude Wind Shear and Its Hazard to Aviation, National Research Council, National Academy Press, 1983.
- 29) Airman's Information Manual, Federal Aviation Administration, 1989.
- 30) "Windshear/Turbulence Inputs to Flight Simulation and Systems Certification", NASA Conference Publication 2474, NASA CP-2474, May 30, 1984. (N97-25267-N87-25288)
- 31) "Development of a Microburst Turbulence Model for the JAWS Wind Shear Data", Ho-Pen Chang, Walter Frost, DOT/FAA/PM-87/12, March 1987.
- 32) "Development and Testing of The Gust Front Algorithm", Arthur Witt, Steven D. Smith, DOT/FAA/PS-87/4, November 1987.
- 33) "A Method for Three-Dimensional Modeling of Wind-Shear Environments for Flight Simulator Applications", Richard S. Bray, NASA TM-85969, July 1984.
- 34) "Wind Shear Detection, Forward-Looking Sensor Technology, NASA Conference Publication 10004, October 1987.
- 35) "A High Resolution Spatial and Temporal Multiple Doppler Analysis of a Microburst and its Application to Aircraft Flight Simulation", Walter Frost et al, DOT/FAA/PM-87/11, March 1987.
- 36) "Integrated FAA Wind Shear Program Plan", DOT/FAA/DL-87/1, April 1987.
- 37) "Airborne Wind Shear Detection and Warning-Systems, First Combined Manufacturers' and Technologists Conference", NASA Conference Publication 10006, January 1988.
- 38) "A Simplified Model of the Turbulent Microburst", L. Roberts and Tung Wan, JLA-TR-59, N86-29466, March 1985.
- 39) "An Examination of a Simulated Microburst Flow As Sensed By a Single Doppler Radar", Eleanor Lee Smith, AFTT/CI/NR 88-90, AD-A196 381, July 1988.
- 40) "The Effect of Heavy Rain on an Airfoil at High Lift", NASA CR 178248, N87-20232, March 1987.
- 41) "Analysis of Doppler Lidar Wind Measurements", NASA 8 34770, N86-29465, May 1986.
- 42) "Augmenting Flight Simulator Motion Response to Turbulence", Lloyd Reid, Journal of Aircraft, April 1990.
- 43) "The simulation of three-dimensional convective storm dynamics". Klemp, J.B., and Wilhelmson, R.B. (1978a) Journal of Atmospheric Science. 35. 1070-1096.
- 44) "Simulations of right and left moving storms through storm splitting", Klemp, J.B. and Wilhelmson, R.B. (1978b). Journal of Atmospheric Science. 35. 1097-1100.
- 45) "Observed and numerically simulated structure of a mature supercell thunderstorm", Klemp, J.B. and Wilhelmson, R.B., (1981) Journal of Atmospheric Science. 38, 1558-1580.
- 46) "Numerical simulation of the evolution of a three dimensional field of cumulus clouds. Part I: Model description, comparison with observations and sensitivity studies", Smolarkiewicz, P.K., and Clark, T.L., (1985) Journal of Atmospheric Science. 42 502-522.
- 47) "A three-dimensional model of cumulus cloud development", Steiner, J.T., (1973) Journal of Atmospheric Science. 30. 414-435.
- 48) "A simulation of the development of successive cells along a cold outflow boundary", Wilhelmson, R.B., and Chen, C.S., (1982) Journal of Atmospheric Science. 39. 1466-1483.
- 49) "Numerical simulation of a cumulus ensemble in three dimensions", Yau, M.K. and Michaud, R., (1982) Journal of Atmospheric Science. 39. 1062-1079.
- 50) "Numerical simulations of an isolated microburst. Part I: Dynamics and structure", Proctor, F.H. (1988) Journal of Atmospheric Science.
- 51) The Weather Wizards Cloud Book, Louis D. Rubin and Jim Duncan, Algonquin Books of Chapel Hill, 1989
- 52) A Short Course in Cloud Physics, R.R. Rogers and M.K. Yau, Pergamon Press, 1989
- 53) FAA Advisory Circular No. 00-54."Pilot Windshear Guide". November 25, 1988.

- 54) FAA Advisory Circular No. 120-40A. "Airplane Simulator and Visual System Evaluation". July 31, 1986.
- 55) FAA Advisory Circular No. 00-50A. "Low Level Windshear". January 23, 1979.
- 56) All About Lighting, Martin Uman, Dover Publications, 1986.
- 57) IVEX, VDS-2000 Specification, 1989.
- 58) Evans & Sutherland, SPX Technical Specification, 1988.
- 59) Windshear Training Aid Volume I, Federal Aviation Administration, February 1987.
- 60) Windshear Training Aid Volume II, Federal Aviation Administration, February 1987.
- 61) Minimum Operational Performance Standards for Airborne Thunderstorm Detection Equipment, RTCA No. RTCA-DO-191, May, 1986.
- 62) "Outside a Cumulus and How to Stay There", Private Pilot, July, 1990.
- 63) Engineering Data Compendium—Human Perception and Performance, 1988.
- 64) Federal Airline Regulations, Part 121, 1990.
- 65) The Terminal Area Simulation System, Fred Proctor, NASA CR 4046, 1987.

ADAPTIVE SIMULATOR MOTION SOFTWARE WITH SUPERVISORY CONTROL

M. A. Nahon* L. D. Reid † J. Kirdeikis

Institute for Aerospace Studies
University of Toronto
Downsview, Ontario, Canada. M3H 5T6

Abstract

Concepts for flight simulator motion-drive algorithms range from the most basic to the relatively complex, with little to guide a choice between these, short of implementing them all and choosing the best. In order to avoid this lengthy process and to put into practice the experience gained in a previous large-scale evaluation exercise, a flexible motion algorithm has been implemented. It can be run as a simple classical algorithm with few free parameters, and can be quickly adjusted to yield good motion performance. The more sophisticated adaptive features of the algorithm can then be brought in gradually to improve performance. Various forms of cost functions and adaptive features were investigated. These were tested on a synergistic 6 degrees-of-freedom motion-base simulator and promising features were identified. Finally, a supervisory code was included to ease motion adjustment, and to provide a safe interactive interface with the designer, as well as automatic motion adjustment to different flight conditions.

Nomenclature

\mathbf{a}_{AA}	Body-axis components of the aircraft acceleration at the cockpit reference pt.
$\mathbf{a}_{SI}, \quad S_I$	Inertial components of the simulator reference point acceleration and pos'n
\mathbf{a}_c	Intermediate acceleration variable in all washout filters
\mathbf{f}_{AA}	Body-axis components of the aircraft specific force at the cockpit reference point ($\mathbf{f}_{AA} = \mathbf{a}_{AA} - \mathbf{g}_A$)
$\mathbf{g}_A, \quad \mathbf{g}_I$	Gravitational acceleration with components in body and inertial frames
G_{x1}, \dots, G_{x4}	Steepest descent slopes in the hybrid algorithm cost function
K_{x1}, \dots, K_{x3}	Fixed coefficients in the hybrid algorithm
\mathbf{L}_{IS}	Rotation matrix transforming vector components from the simulator reference frame to the inertial frame
\mathbf{p}'	Adaptive coefficients in the coordinated adaptive algorithm

*Student Member, AIAA

[†]Associate Fellow, AIAA

P_{x1}, \dots, P_{x4}	Adaptive coefficients in the hybrid algorithm
P_{x10}, \dots, P_{x40}	Reference values of the adaptive coefficients in the hybrid algorithm
s	Laplace operator
T_S	Transformation matrix from angular velocity to Euler angle rates
W_{x0}, \dots, W_{x9}	Weights in the hybrid algorithm cost function
β_A, β_S	Aircraft and simulator Euler angles ($\beta = [\phi \ \theta \ \psi]^T$)
ω_{AA}	Body-axis components of the aircraft angular velocity

Vectors and matrices are denoted by boldface type.

Introduction

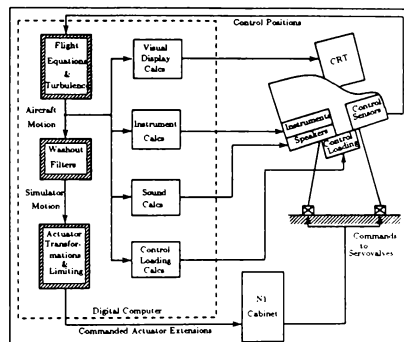


Fig. 1. A Typical Flight Simulator installation

A typical flight simulator installation is made up of the subsystems shown in Figure 1. Of these, the ones which affect the quality of the flight simulator motion have been highlighted: the flight/turbulence model; the 'washout' algorithm; and the soft-limiting system. The washout filters (whose name originates from the fact that one of their functions is to 'wash out' the position of the simulator back to its neutral point) remain a primary source of poor fidelity in the motion cues, and even a small improvement in this area can yield

a significant improvement in motion realism. Briefly stated, the purpose of the washout subsystem is to take the motions generated by the aircraft equations of motion—which include very large displacements—and filter them to provide simulator motion-base commands. These commands must provide the pilot with realistic motion cues, while remaining within the simulator's motion limits. Various techniques have been proposed to achieve this effect.

One of the earliest and simplest types is the classical washout in its many forms^{1,2,3} generally characterized by mostly linear elements assembled by trial and error. Its principal advantages are its simplicity and ease of adjustment. Adaptive schemes were later proposed in which the filter gains are altered in *real time* in order to minimize a cost function using steepest descent techniques.^{4,5,6} Since the cost function contains simulator excursions from the neutral point as well as motion errors, smaller inputs should be attenuated less than larger ones since they are less likely to result in large simulator displacements. Thus, false cueing should be reduced and better use made of the motion-base capabilities. More recently, optimal control theory has been applied to the problem^{7,8} to generate an optimal linear filter which would minimize a specified quadratic cost function *off-line*.

The work of References 9 to 11 summarized in References 12 and 13 presented a comparison of representative classical, adaptive and optimal algorithms undertaken with the aim of choosing one of these for the University of Toronto Institute for Aerospace Studies (UTIAS) Flight Research Simulator. Results of that work showed that, on balance, the classical and adaptive algorithms were better suited to our application.¹² Further work has since been directed at combining the best features of the classical and adaptive schemes into a more comprehensive *hybrid* scheme. This approach was originally proposed in Reference 14 though few details were given there of the implementation. The assumption of decoupled translational, rotational and crossfeed filters used in that work could lead to false cueing if the rotational and crossfeed channels both try to generate a given aircraft motion.

In the present work, nonlinear adaptive filter blocks have been incorporated in the classical layout. The choice of layout, as well as the location of the adaptive filter blocks is justified step-by-step to show that this hybrid washout filter should combine the best features of the classical and adaptive algorithms with few of their disadvantages. Alternative cost functions were investigated in an attempt to enhance the adaptive features of the algorithm. These included second, fourth and sixth order, as well as piecewise linear formulations. The adaptive filter equations were also extended to add adaptive frequency and damping features to the filters to see whether these would be more effective than

the more traditional adaptive gain. The performance of these features was first evaluated off-line in order to determine their resistance to instabilities, their effect on ease of adjustment and their improvement of the motion cues. The most promising algorithms were then implemented and evaluated on the UTIAS Flight Research Simulator in an informal pilot evaluation.

All washout algorithms contain a number of free parameters which must be adjusted by trial and error to produce the desired motion. In the process of implementing this and previous algorithms, it was noted that the parameter values critically affect an algorithm's performance. Because so much time was spent adjusting these parameters, a facility to ease this *tuning* process was implemented. This supervisory software enabled the designer to modify the motion of the simulator in consultation with the evaluation pilot in a smooth interactive manner. This has significantly reduced turnaround time and enabled the pilot to experience different motions during simulator 'flight' in quick back-to-back succession for easier comparison. A number of safety features were incorporated in order to ensure that changes in the motion parameters did not cause bumps or instabilities. Finally, since the washout filter parameters necessary for realistic motion differ according to the flight phase (e.g., they are different for ground handling and for flight), the supervisory software was extended to allow automatic changes of the filter parameters according to certain control flags. This feature could also be used to cater to differing pilot preferences in simulator motion.

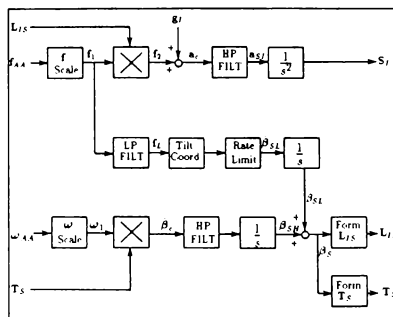


Fig. 2. The Classical Algorithm

Layout

In this section the similarities between the classical and adaptive algorithms will be highlighted in support of including these same features in the new hybrid algorithm. The principal differences between the two algorithms will then be discussed and justification will be given for choosing one approach or the other for the new algorithm. Figures 2 and 3 show the flowcharts of

the classical and adaptive algorithms as they were implemented in References 9 to 13. Their similarities are as follows:

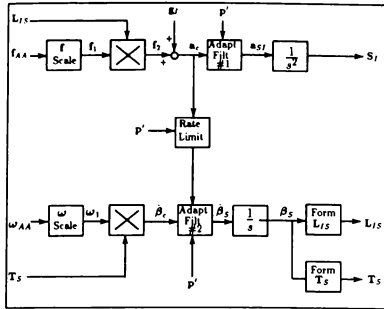


Fig. 3. The Adaptive Algorithm

- 1) The inputs to the washout filters are the aircraft specific forces (f_{AA}) and angular rates (ω_{AA}). Although other motion variables could be used as inputs, the objective of the washout filter is inherently to reproduce f_{AA} and ω_{AA} ; a task more easily accomplished by using these as inputs from the outset. Although some authors have suggested using sensed f_{AA} and ω_{AA} as the variables which should be reproduced⁸, our evaluation of this suggestion has shown that the benefits do not warrant the added CPU time.⁹

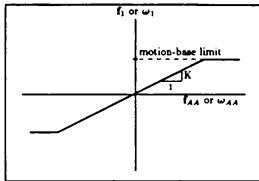


Fig. 4. Scaling & Limiting

- 2) The motion inputs (f_{AA} and ω_{AA}) are scaled and limited according to Figure 4 (producing f_1 and ω_1) in order to reduce the amplitude of the motions to be simulated. Input scaling renders the task of filtering easier by subjecting all input motions to attenuation at all frequencies. It was found that scale factors, K , of between 0.5 and 1.0 were liked best by pilots (depending on the pilot, the degree-of-freedom and the flight maneuver). Input limiting serves as a safety feature by ensuring that excessive commands do not enter the washout filter. In the present implementation, these were set at experimentally-determined motion-base limits in acceleration and velocity.
- 3) The scaled and limited body-axis aircraft motion

(represented by f_1 and ω_1) are premultiplied by $L_{I/S}$ and T_S to obtain f_2 and β_2 , respectively. This allows the subsequent filtering to be performed in an inertial frame. As discussed in Reference 9, filtering can be performed in the body frame, the inertial frame or some combination of the two. Inertial-frame filtering is advantageous because it ensures that all simulator motions will be washed out to zero (assuming filters of a sufficiently high order), although it introduces some small second-order coupling effects. Body-frame filtering does not introduce this coupling, but does not ensure that the motion-base motions will be washed out. Test cases were run in Reference 9 which showed that the cross-coupling effect was negligible compared to the problem of large offsets when the motion was not properly washed out. As a result, all filtering in the hybrid algorithm is performed in the inertial frame.

- 4) The transformed aircraft translational acceleration, represented by a_x , is passed through high-pass filters to obtain the simulator translational acceleration, a_{SI} . Low-frequency inputs to the washout filters will tend to generate excessive simulator displacement, whereas high-frequency signals (such as those resulting from turbulence) will, in general, be reproducible. In order to separate the high and low frequency components, the translational accelerations are passed through high-pass filters. The filter output is then integrated twice to produce the simulator translational position, S_I .
- 5) The above filtering will only yield the high-frequency content of the aircraft translational motion. Low-frequency aircraft translational motion is simulated using 'tilt-coordination', or crossfeed, from aircraft translational motion to simulator rotational motion. This mechanism accounts for a great deal of the realism possible in a simulator, and makes the motion simulation of large transport aircraft substantially easier than that of fighter aircraft (since the frequency content is lower for the former than the latter). Both the classical and adaptive algorithms include a form of crossfeed, although the detailed implementation differs (this will be discussed later in more detail). Furthermore, in order to prevent the pilot from noticing this 'trick', tilt-rate limiting is used on the crossfeed channels. The tilt-rate limits used in all the algorithms considered here were 3 deg/s in pitch and 2 deg/s in roll, consistent with the findings of a previous study.¹¹
- 6) The outputs from the washout filter are S_I and β_S , the translational and angular positions of the motion-base. These are processed through actuator transformation⁹ and soft-limiting¹⁰ algorithms to obtain commanded actuator lengths. Finally, the simulator Euler angles β_S are used to form the transformation matrices $L_{I/S}$ and T_S to be used in

the next pass through the washout algorithm.

There are also some substantial differences between the classical and adaptive washout algorithms. These are now enumerated and an argument is made as to which of the two approaches should be used in the hybrid algorithm.

- 1) The primary difference between the adaptive and classical algorithms is their filter gains. Whereas the classical filters have fixed gains (of 1.0 in the present case), the gain of the adaptive filters varies during 'flight' in an attempt to reduce a cost function. The cost function is composed of motion 'errors' (difference between simulator and aircraft motion) and simulator displacements and velocities. This feature gives the adaptive algorithm a certain amount of 'intelligence' which allows it to reduce 'false cueing' and to attenuate motion commands which would tend to drive the motion-base into its limits. Adaptiveness was therefore included in the hybrid algorithm.
- 2) The location of the crossfeed relative to the L_{JS} transformation is different in the two algorithms. In the classical washout algorithm, the crossfeed signal is taken before the transformation, while in the adaptive algorithm, it is taken after. This has an important effect on when the crossfeed channel is active. In the classical algorithm, the crossfeed will be active only when a body-axis longitudinal or lateral specific force is present in the aircraft, while in the adaptive algorithm, the crossfeed is active over a much broader range of situations. For example, in a coordinated turn (a maneuver during which the classical crossfeed remains inactive), the aircraft body-axis vertical specific force will be fed to the lateral crossfeed channel via the L_{JS} transformation in the adaptive algorithm. In fact, this mechanism is responsible for cancelling the low-frequency angular rate, since the adaptive algorithm does not have explicit filtering of roll and pitch. Because the adaptive crossfeed channel is more active than the classical crossfeed, its adjustment is more complex since it entails finding a compromise in its behavior over a wider range of maneuvers. The classical crossfeed is much easier to adjust since it performs a simpler task. For these reasons, the hybrid algorithm used the crossfeed layout found in the classical algorithm.

- 3) Roll and pitch motions are explicitly high-pass filtered in the classical algorithm, while the adaptive algorithm relies on the crossfeed channel to cancel low-frequency angular motion. The final behaviour of both systems is to let uncoordinated aircraft rotational motions through unattenuated at all frequencies, while acting as a high-pass filter for coordinated aircraft rotational motions.¹⁰ The question of whether or not to include explicit high-pass filtering on these channels is intrinsically tied to the location

of the crossfeed. Once it was decided to take the crossfeed from upstream of the L_{JS} transformation we were constrained to include explicit rotational filters in the hybrid algorithm.

- 4) The crossfeed channel in the adaptive algorithm feeds directly into the angular rate channel, while in the classical algorithm, it feeds through a low-pass filter into angular position. The difference in these approaches is again a result of the location of the source of the crossfeed signal relative to the L_{JS} transformation. Therefore, the form of crossfeed chosen was the same as that in the classical algorithm.

Although adaptive filters are to be included in the new hybrid algorithm, not *all* of the filter blocks need to be adaptive. The hybrid algorithm includes the following filters:

- High-pass filters on all three translational channels (second-order in the longitudinal and lateral directions, and third-order in the vertical direction),
- High-pass filters on all three rotational channels (first-order in pitch and roll, and second-order in yaw),
- Low-pass filters on the pitch and roll crossfeed channels (second-order).

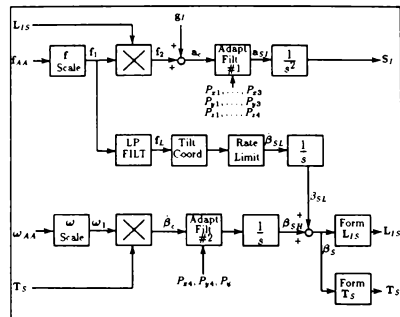


Fig. 5. The Hybrid Algorithm

Of these, the first two sets were made adaptive, while the last was left fixed. The justification for this was that the adaptive nature of the crossfeed in the original adaptive algorithm led to a great deal of trouble in tuning while yielding few improvements in motion realism. Furthermore, since the output of the crossfeed channel is meant to 'trick' the pilot, it is important to know at all times when and why this channel is active, and to keep strict limits on this activity. This requires crossfeed of the simple fixed form found in the original classical washout.

From the preceding discussion, the layout shown in Figure 5 was obtained for the new algorithm. It is identical to that of the classical filter except that certain filter blocks can be made adaptive. One of the advantages of this layout is that the degree of adaptiveness can be limited as desired, and it is possible to reduce it to a classical algorithm which can be fairly easily tuned to obtain good performance. From this point, adaptiveness can be introduced gradually, one filter at a time, to improve the performance.

Form of the Adaptive Filters

The original adaptive washout algorithm developed in References 4 and 5 used a quadratic cost function to vary the gain of the filters in real time. The present work investigates the use of higher-order cost functions as well as the use of adaptive break frequency and damping to provide a more versatile adaptive filter. The basic adaptive filter equations implemented in the present work will now be reviewed in more detail. The pitch/surge equations are given as an example:

$$\begin{aligned} \alpha_c^x &= f_1^x \cos \theta_S \cos \psi_S \\ &+ f_2^x (\sin \phi_S \sin \theta_S \cos \psi_S - \cos \phi_S \sin \psi_S) \\ &+ f_3^x (\cos \phi_S \sin \theta_S \cos \psi_S + \sin \phi_S \sin \psi_S) \quad (1a) \\ \dot{\theta}_c &= q_1 \cos \phi_S - r_1 \sin \phi_S \quad (1b) \end{aligned}$$

$$\ddot{\theta}_c^x = P_{x1} \alpha_c^x - P_{x2} \dot{\theta}_c^x - P_{x3} \dot{\theta}_c^x \quad (2a)$$

$$\ddot{\theta}_{SL} = K_{x1} \frac{\ddot{\theta}_c^x}{g} - K_{x1} \theta_{SL} - K_{x2} \dot{\theta}_{SL} \quad (2b)$$

$$\ddot{\theta}_{SH} = P_{x4} \dot{\theta}_c - K_{x3} \theta_{SH} \quad (2c)$$

$$\dot{\theta}_S = LIM(\dot{\theta}_{SL}) + \dot{\theta}_{SH} \quad (2d)$$

The cost function is given by

$$\begin{aligned} J_x &= \frac{1}{2^{N-1}} [W_{x0} (\alpha_c^x - \ddot{\theta}_c^x)^N + W_{x1} (\dot{\theta}_c - \dot{\theta}_S)^N \\ &+ W_{x2} (\ddot{\theta}_c^x)^N + W_{x3} (\dot{\theta}_c^x)^N + W_{x4} (\dot{\theta}_S)^N + W_{x5} (\theta_S)^N \\ &+ W_{x6} (P_{x1} - P_{x10})^N + W_{x7} (P_{x2} - P_{x20})^N \\ &+ W_{x8} (P_{x3} - P_{x30})^N + W_{x9} (P_{x4} - P_{x40})^N] \quad (3) \end{aligned}$$

The P_{xi} coefficients in equation (2) are varied in real time according to

$$\dot{P}_{xi} = -G_{xi} \frac{\partial J_x}{\partial P_{xi}} \quad i = 1, \dots, 4 \quad (4)$$

Equation (2a) represents the second-order adaptive high-pass longitudinal filter; equation (2b) represents the fixed second-order low-pass crossfeed filter; while equation (2c) represents the first-order adaptive high-pass pitch filter. The P_{xi} parameters are adaptive: P_{x1} is the gain, P_{x2} is the square of the break frequency, and P_{x3} is twice the product of the filter damping ratio and break frequency, all for the translational filter. P_{x4}

is the adaptive gain of the pitch high-pass filter. The K_{xi} parameters are fixed: K_{x1} and K_{x2} are the square of the break frequency and twice the product of break frequency and damping ratio of the crossfeed filter, respectively, while K_{x3} represents the break frequency of the rotational high-pass filter. Adaptive break frequencies were not included in the rotational filters as it was felt sufficient to evaluate this feature on the translational channels. Had the performance of this feature been satisfactory, it could have been implemented in the rotational channels as well.

Equations (1), (2) and (3) can be substituted into Equation (4) to obtain P_{xi} ($i = 1, \dots, 4$), as explicit functions of the input variables f_1 and ω_1 . These relations can then be integrated in real-time to obtain the values of the adaptive parameters, P_{xi} ($i = 1, \dots, 4$). All integrations in the digital implementation of the washout algorithm were performed using the (first order) 'ABC' method—a variation on the Euler method which uses updated values of variables as they become available.⁹ This was done to avoid introducing lags in the filter response which are characteristic of the explicit Euler method. If computing power were a lesser constraint, a higher order implicit method would be preferred.

Varying the adaptive parameters in real time according to equation (4) will tend to minimize the cost function J_x given by equation (3). The first two terms in the cost function penalize motion error in x-acceleration and in pitch rate; the next four terms penalize the simulator motion; while the last four terms are included to return the adaptive parameters to their original reference state. As a safety measure to avoid instabilities in the adaptive parameters which develop when they vary too much, the values P_{xi} ($i = 1, \dots, 4$) were passed through limiting blocks to ensure they remained within reasonable bounds. These bounds were established for each parameter as a function of its reference value (e.g., $0 \leq P_{x1} \leq P_{x10}$).

An important feature of the hybrid algorithm is that it becomes identical to the original classical algorithm when the steepest descent slopes G_{xi} ($i = 1, \dots, 4$) are set to zero. In this case, the parameters W_{xi} ($i = 0, \dots, 9$) and P_{x10} ($i = 1, \dots, 4$) no longer affect the simulator motion, and the designer need only adjust K_{xi} ($i = 1, \dots, 3$) as best possible—a task known to be relatively easy.¹³ If better performance is required, the steepest descent slopes can be gradually increased while choosing less restrictive values for K_{xi} ($i = 1, \dots, 3$) and adjusting the cost function weights as needed. It should also be noted that selectively setting certain steepest descent slopes to zero will alter the resulting type of adaptive filter. Referring, for example, to Equations (2) to (4), if G_{x2} and G_{x3} are set to zero while G_{x1} and G_{x4} are non-zero, the resulting filters will have adaptive gain and fixed break frequencies and

damping. This feature allowed us to evaluate various combinations of adaptive and fixed parameters.

Different values of the order of the cost function, N in equation (3), were also evaluated: 2, 4, 6 and piecewise linear with varying slopes as shown in Figure 6. The piecewise-linear cost function was composed of 100 linear segments over the range $-2 < x < 2$ (with constant slope outside this range) to produce the shape shown in Figure 6. It was conjectured that cost functions which were flatter near the origin and rose more quickly as the independent variable approached a 'critical value' (i.e., the motion limits), would show greater tendency to attenuate large motions more than small ones.

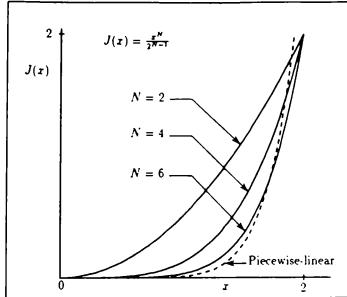


Fig. 6. Order of the Cost Function

For each order of the cost function, three parameter sets were evaluated: 1) G: Adaptive gain only on translational and rotational channels (i.e., $G_{x1} \neq 0$, $G_{x2} = G_{x3} = 0$, $G_{x4} \neq 0$); 2) F: Fixed gain and adaptive break frequency and damping on the translational channels, and adaptive gain on the rotational channels (i.e., $G_{x1} = 0$, $G_{x2} \neq 0$, $G_{x3} \neq 0$, $G_{x4} = 0$); 3) GF: All parameters adaptive (i.e., $G_{x1} \neq 0$, $G_{x2} \neq 0$, $G_{x3} \neq 0$, $G_{x4} \neq 0$). Thus, in all, twelve different combinations of characteristics were evaluated. These were denoted as AW2G, AW2F, AW2GF, AW4G, AW4F, AW4GF, AW6G, AW6F, AW6GF, AWPLG, AWPLF and AWPLGF, where AW denotes an adaptive algorithm, 2, 4, 6 and PL denote the order of the cost function, and G, F and GF denote which filter parameters allowed to adapt.

Testing the Algorithm

The evaluation of the various forms of the adaptive filters was performed in three phases:

- 1) An off-line assessment of the motion response to idealized single degree-of-freedom motion inputs consisting of: a) a double pulse in longitudinal acceleration, b) a double pulse in vertical acceleration, c) an uncoordinated sinusoidal roll motion and d) a ramp up to a steady value of yaw rate.

- 2) An off-line evaluation of the motion response to pre-recorded aircraft maneuvers. These motion inputs were recorded while a pilot flew a simulated Boeing 747 through the two following maneuvers: a) a series of three alternating turn entries and b) a pushover/pull-up maneuver.
- 3) An informal piloted assessment during which an evaluation pilot flew the simulated Boeing 747 on the UTIAS Flight Research Simulator described in Reference 11. The pilot was rated on light aircraft and had considerable experience in the simulator. The maneuvers flown included a) coordinated turn entries, b) pushover/pull-up, c) sideslip, d) approach through turbulence and e) touch and go landing.

The results of the first two phases of testing showed the following:¹⁵

- 1) The filters with adaptive gain only (suffix G) tended to adapt more rapidly than the filters which included adaptive break frequency and damping parameters (F and GF suffix),
- 2) The filters with adaptive break frequency and damping (suffix F) were not as effective at limiting maximum simulator displacements as the adaptive gain filters (suffix G). This dictated the use of more restrictive filter characteristics (larger values of P_{x20} and P_{x30}) to prevent the motion from exceeding its limits,
- 3) The filters with adaptive gain and break frequency and damping parameters (suffix GF) behaved midway between the filters with adaptive gain only (suffix G) and those with adaptive break frequencies and damping only (suffix F). However, the effect of adaptive gains seemed to predominate over that of the adaptive break frequencies and damping,
- 4) The second-order filters adapted more rapidly than the sixth-order filters,
- 5) The response of the piecewise-linear filters was very similar to that of the continuous quadratic formulation but the motion produced was not as smooth,
- 6) The piecewise-linear filters sometimes exhibited high-frequency oscillations in their motion output as the slope toggled between two discrete values. For example, Figure 7(a) shows a high-frequency ringing in the lateral acceleration response of the AWPLG filter to the idealized roll input. By contrast, Figure 7(b) shows the smooth response obtained with the AW4G filter.

Following the first two phases of evaluation, seven of the candidate formulations were eliminated from further testing as they showed no benefits over the other formulations. These included all three formulations with piecewise-linear cost functions as they produced

frequent oscillations and discontinuities due to changes in the slope of the adaptive parameters. The AW4F, AW4GF, AW6F and AW6GF formulations were discarded since they showed little or no adaptive response. Thus, in the third phase the AW2G, AW2F, AW2GF, AW4G and AW6G formulations were compared to each other and to the standard classical algorithm shown in Figure 2. The pilot comments were as follows:

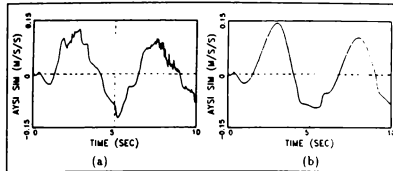


Fig. 7. Response to Idealized Roll Input

- CW: The classical washout provided good smooth motion with no noticeable phase lags. However, most motion cues were very attenuated and the actuator displacement limits were hit during the pull-up maneuver.
- AW2G: This formulation provided stronger motion cues which were well coordinated with control inputs and the visual display. The touchdown bump was more realistic as were the motion cues during approach. The washout (motion of the moving platform returning to its neutral position) was more noticeable than in the classical washout but was not objectionable. The actuator limits were not reached.
- AW2F: The pilot strongly disliked the motion produced by this algorithm. It produced noticeable phase lags and frequent false cues which were disorienting. The washout motion was perceptible and jerky and the actuator limits were reached during the pull-up maneuver.
- AW2GF: The motion produced by this formulation was midway between AW2G and AW2F. Some phase lags were apparent during the turn entries and some false cues were perceptible. The pilot compared the touchdown bump to landing on a sponge. The actuator limits were reached during the pull-up maneuver.
- AW4G: This formulation elicited the most favourable comments. The turn entries produced a strong rolling sensation which was in phase with the visuals; the sideslip cues were realistic and well coordinated and the touchdown bump was convincing. The actuator limits were not hit. The washout motion was perceptible but not jerky.
- AW6G: This formulation produced strong onset cues but the pilot found the washout motion too strong, particularly during turn entries. The actuator limits were reached during the pullup maneuver and some

pitch oscillations were experienced.

Summarizing the preceding comments, it was found that the formulations were rated from best to worst as AW4G, AW2G, CW, AW6G, AW2GF and AW2F. The adaptive frequency and damping parameters did not do a good job of avoiding the actuator limits and introduced significant phase lags which the pilot found disorienting. The AW2G and AW4G formulations provided stronger cues than the classical algorithm while also doing a better job of avoiding the actuator limits. Referring back to the governing equations (2) to (4), the P_{x2} and P_{x3} adaptive parameters could be replaced by fixed parameters, thereby reducing the number of parameters to be specified (i.e., G_{x2} , G_{x3} , P_{x20} and P_{x30} could be dropped) and simplifying the resulting equations.

Supervisory Control

The adaptive filter equations include a number of parameters whose values must be chosen by the designer. For example, the pitch/surge filter given by Equations (2) to (4) is adjusted by choosing W_{xi} ($i = 0, \dots, 9$), G_{xi} ($i = 1, \dots, 4$), P_{xi0} ($i = 1, \dots, 4$), and K_{xi} ($i = 1, \dots, 3$). The values chosen for these parameters will strongly affect the motion produced by the washout algorithm for a given aircraft motion. The adjustment of these parameters is commonly referred to as motion tuning or adjustment—it is an interactive exercise where a test pilot flies the simulator, comments on the motion which he experiences and the designer modifies the motion parameters accordingly. Since the parameters in any washout algorithm greatly affect the simulator's motion, the tuning process becomes critical. In the present exercise, it was therefore important to facilitate this operation as much as possible. This section describes the software developed to implement and supervise the tuning process, thereby making motion adjustment an integral part of the overall simulation.

Structure of the Simulation Environment

The subsystems shown in Figure 1 are only part of the total simulation environment—those which make up the real-time simulation. In fact, a more complete representation of the total environment is shown in Figure 8, and includes the designer who acts as a high-level controller. In the present implementation, the designer (who may be located inside or outside the simulator cab) is seated at a console which allows him to control a low-priority interactive task in the host computer. A high-priority real-time task coexists in the computer which schedules and executes all the software subsystems shown in Figure 1. Both tasks have access to certain variables in shared memory. These are either information variables transmitted from the real-time task to the designer to inform him of the progress of the simulation (e.g., airspeed, altitude), or control variables which can be altered by the designer or instructor to control

the evolution of the training session (e.g., engine failure flag).

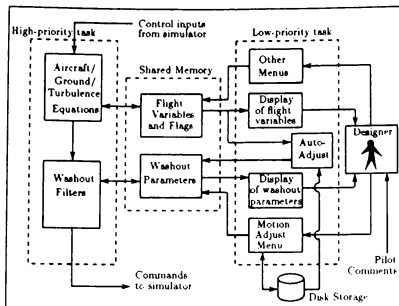


Fig. 8. The Complete Simulation Environment

Since the present application required fast modification of the washout filter parameters without interrupting the flight, these parameters were placed in shared memory and an interactive menu was provided to alter them. The menu has options to display or change any parameter; load a complete parameter set from file or store a complete parameter set to file. Thus the pilot could voice a complaint, the designer would identify which parameters needed alteration and perform the change, and query the pilot for any perceived changes. This efficient setup allowed complete motion tuning within 3 or 4 two-hour sessions (the sessions were kept short to avoid tiring the pilot).

Motion Adjustment

During the flight, the designer interacts with the low-priority task via a menu which allows him to select an action from a list which includes the following motion-related options:

- 1) Change the complete set of washout filter parameters,
- 2) Display/Adjust individual washout filter parameters,
- 3) Store the complete set of washout filter parameters.

The above menu items are given in more detail in Figure 9. As shown in that figure, a change of washout filter entailed reading a new set of filter parameters from a file and replacing the old filter parameters with these. In the filter adjustment option, the designer chooses a specific parameter, its present value is displayed, and the designer is prompted for the new desired value. Once any of these changes has been made, the value of the appropriate parameter in shared memory is changed, and the change takes effect in the real-time simulation. Once the motion adjustment session is completed, all the parameter values can be read

from shared memory and stored in a file.

Since large instantaneous changes in certain washout filter parameters could result in bumps or instabilities in the motion, a rate-limiting algorithm was implemented to filter out these discontinuities as follows:

$$\dot{x} = (x_{in}^i - x_{out}^{i-1})/\delta t \quad (5a)$$

$$x_{out} = LIM(\dot{x}) \quad (5b)$$

$$x_{out}^i = x_{out}^{i-1} + \delta t \cdot \dot{x}_{out} \quad (5c)$$

where x_{in} represents the input variable, x_{out} represents the rate-limited variable, and the i and $i-1$ superscripts denote values at the present and previous time steps, respectively. This algorithm ensures that the output is identical to the input as long as the rate limit is not reached, and that even after rate limit has occurred, the output will eventually resume tracking the input. Preset limits were set for every variable, and the entry console emitted an auditory cue as long as any parameter was being rate-limited in order to notify the designer that the change just made was not yet fully in effect.

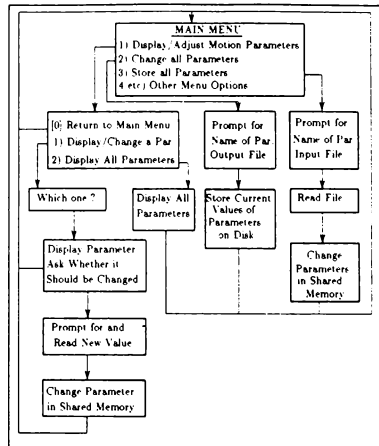


Fig. 9. Motion Adjustment Flowchart

A software emergency shutdown sequence was also provided as an additional safety feature. Hitting the return key in any sub-menu without entering a response would return control to the main menu. Repeating this action from the main menu would shut down the simulator motion without stopping the rest of the simulation. Thus, when a motion instability seemed incipient, the designer could halt all motion by simply hitting the return key twice. Of course, the designer also had the usual hardware panic button for more severe emergencies, though its use entailed a more complex restart procedure.

Extension to Automatic Motion Adjustment

One of the interesting results of the motion tuning was that the set of motion parameters which produced the best motion changed with the flight phase. A dramatic example of this is the substantial difference in the motion parameters needed for realistic motion in flight and during ground maneuvering. The software described above was therefore extended to provide automatic changes in parameters for different phases of flight according to (for example) the touchdown flag, the turbulence flag etc. Thus, these changes were not made by the instructor through the console, but rather by the appropriate flag changing in the real-time task, being communicated to the low-priority task, which would then summon automatic versions of the washout parameter change routine. The rate-limiting algorithm mentioned in the previous section ensured that all transitions occurred smoothly and went unnoticed by the pilot.

A further application of automatic motion adjustment pertained to the inter-pilot variability in motion assessment encountered during the piloted evaluations.¹¹ Different pilots have distinctly different perceptions of what type of motion is most realistic. These subjective preferences could not be satisfied with a single parameter set. Rather, each pilot could be assigned a file containing his preferred parameter set which would be invoked at the start of a simulator session. Of course, further investigation would be needed in order to determine whether the transfer of training is improved or degraded by catering to subjective pilot preferences.

Conclusions

A flexible motion system has been developed for use on moving-base simulators. It consists of a washout algorithm which combines the best features of the common existing algorithms, controlled by supervisory software. The new algorithm was derived as a hybrid of existing classical and adaptive algorithms. This allowed a filter which could be easily adjusted to obtain good performance, while adaptive features could be added as required. Various forms of the cost function were investigated and it was found that the fourth-order cost function with adaptive-gain filters received the most favourable response. Adaptive break frequency and damping were also investigated, but they were found to produce objectionable phase lags in the motion and did not do a good job of avoiding the actuator limits. A supervisory software system was constructed to allow fast interactive testing and adjustment of motion algorithms. It was also extended to include automatic changes according to flight phase and conditions, or according to pilot preference.

Acknowledgements

The work reported here was funded jointly by

AERCOL Ltd. and the Natural Sciences and Engineering Research Council. The development of the simulator used in this study was funded by the Canadian Natural Sciences and Engineering Research Council and the Ontario Government.

References

- ¹ Schmidt, S. F. and Conrad, B., "Motion Drive Signals for Piloted Flight Simulators," NASA CR-1601, May 1970.
- ² Baarspul, M., "The generation of Motion Cues on a Six-Degrees-of-Freedom Motion System," Delft University of Technology, Dept. of Aerospace Engineering, Report LR-248, June 1977.
- ³ Parrish, R. V., Dieudonne, J. E. and Martin, D. J. Jr., "Motion Software for a Synergistic Six-Degree-of-Freedom Motion Base," NASA TN D-7350, Dec. 1973.
- ⁴ Parrish, R. V., Dieudonne, J. E., Bowles, R. L. and Martin, D. J. Jr., "Coordinated Adaptive Washout for Motion Simulators," *AIAA Journal of Aircraft*, Vol. 12, No. 1, Jan. 1975, pp. 44-50.
- ⁵ Parrish, R. V. and Martin, D. J. Jr., "Comparison of a Linear and Nonlinear Washout for Motion Simulators Utilizing Objective and Subjective Data from CTOL Transport Landing Approaches," NASA TN D-8157, June 1976.
- ⁶ Ariel, D. and Sivan, R., "False Cue Reduction in Moving Flight Simulators," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-14, No. 4, July/Aug. 1984, pp. 665-671.
- ⁷ Sivan, R., Ish-Shalom, J. and Huang, J.-K., "An Optimal Control Approach to the Design of Moving Flight Simulators," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-12, No. 6, Nov./Dec. 1982, pp. 818-827.
- ⁸ Ish-Shalom, J., "The Design of Optimal Control Motion for Flight Simulators," Ph.D. Thesis, M.I.T. Centre for Space Research, Dec. 1982.
- ⁹ Reid, L. D. and Nahon, M. A., "Flight Simulator Motion-Base Drive Algorithms: Part 1-Developing and Testing the Equations," University of Toronto, UTIAS Report No. 296, Dec. 1985.
- ¹⁰ Reid, L. D. and Nahon, M. A., "Flight Simulator Motion-Base Drive Algorithms: Part 2-Selecting the System Parameters," University of Toronto, UTIAS Report No. 307, May 1986.
- ¹¹ Reid, L. D. and Nahon, M. A., "Flight Simulator Motion-Base Drive Algorithms: Part 3-Pilot Evaluations," University of Toronto, UTIAS Report No. 319, Dec. 1986.
- ¹² Reid, L. D. and Nahon, M. A., "The Response of Airline Pilots to Variations in Flight Simulator Motion Algorithms," *AIAA Journal of Aircraft*, Vol. 25, No. 7, July 1988, pp. 639-646.
- ¹³ Nahon, M. A. and Reid, L. D., "Simulator Motion Drive Algorithms-A Designer's Perspective," *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 13, no. 2, pp. 356-362, March/April 1990.
- ¹⁴ Dorbolo, G. and Van Sliedregt, J. M., "Improvements in Motion Drive Algorithms," *Proceedings of the 1987 Summer Computer Simulation Conference*, July 1987, Montreal, Canada, pp. 721-723.
- ¹⁵ Kirdeikis, J., "Evaluation of Nonlinear Motion-Drive Algorithms for Flight Simulators", UTIAS Technical Note No. 272, June 1989.

MICROPROCESSOR CONTROL FOR A SIMULATOR MOTION PLATFORM

Robert W. Levi*
Robert Levi Associates
Anaheim, CA.
(714) 956-7935

Robert A. Walker* and Travis Wilson
McDonnell Douglas Corp.
Long Beach, CA.

Larry Hayashigawa*
McFadden Systems Inc.
Santa Fe Springs, CA.

Abstract.

A microprocessor embedded into the control system of a simulator motion platform offers many benefits to the motion system user, including ease of interfacing a variety of host simulation computers, reduced host computation loading, and many enhanced operational features. This paper describes the design of such a motion system including hardware and software architecture, resolution and sample rate requirements and analysis, program coding techniques and data structures. Discussions are also presented for the mechanical and hydraulic systems configuration and design.

Introduction.

Douglas Aircraft Co.'s newly rebuilt research and development simulator motion system incorporates a microprocessor controller to interface, control and monitor the motion system. The desire to retain the generic cockpit and the existing 72 inch actuator stroke and geometry dictated a custom designed, six axis synergistic motion control system. The microprocessor controller accomplishes many functions that have traditionally been performed by the host simulation computer, including, degree-of-freedom kinematic calculations and motion drive algorithms, washouts and dynamic leg limiting. The microprocessor also controls the hydraulic pumps, powered stairway and all indicators and annunciators. A redundant electromechanical safety scheme allows the microprocessor to continuously monitor and control the system status via numerous hydraulic system sensors, position transducers, servo error voltages, limit and safety switches, and other sensors.

The interfacing scheme for the host computers is particularly interesting, in that two different types of computers are used for simulation of different aircraft, namely MD-11 and C-17. The motion system provides a nearly transparent interface to the host by requiring only transformed aircraft accelerations; differences in motion requirements for the two aircraft are taken care of by the microprocessor control-

ler. The design of the system allows the cockpit crew full operational control of the motion system without intervention from outside personnel or the host computer. This stand-alone feature simplifies maintenance procedures and reduces support personnel requirements.

Motion System Upgrade for an Existing Simulator.

The newest aircraft from McDonnell Douglas imposed motion simulation requirements on the company's research and development facility that could not be met by the existing system. The original facility was designed and built in the early 1970's using in-house resources and the synergistic hexapod design licensed from The Franklin Institute. 72" stroke actuators were used to provide an ample operating envelope.

The original system served the company well for over 15 years, but increased performance requirements called for an overhaul. To minimize downtime and impact to ongoing projects using the existing simulator cockpit, it was desirable to replace the motion system without disturbing the integrity of the cockpit and the electrical umbilical cables. Also this required the existing geometry to be maintained, so that no mechanical alterations would be required on the platform or the reaction mass attachment points.

Facility Description.

The McDonnell Douglas Crew Station Simulation Laboratory uses two varieties of host simulation computers to model different types of aircraft. Nominal transport aircraft are handled by a VAX 8650. More demanding simulations, such as the C-17, are processed by an ELXSI 6400. A MicroVAX II handles input and output from the active host computer via a fiber-optic data link, thus freeing the host for the primary task of solving the aircraft equations of motion.

Other components of the flight simulator include a generalized wide-body transport cockpit, a dual station McFadden control force loading system, a cockpit noise and engine sound system, and a Redifon Visual Flight Attachment for forward window visual cues. Visual images are furnished by

* Member, AIAA

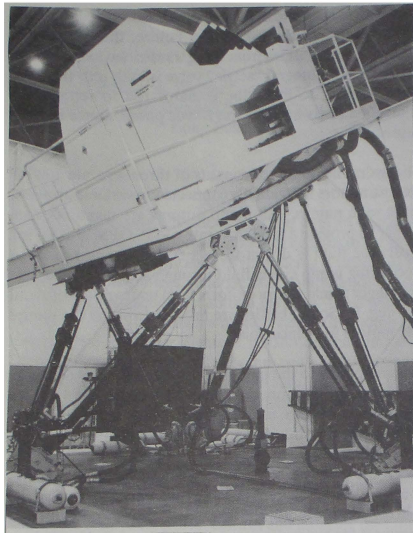


Figure 2. Motion Base Installation

Performs an array of self test operations at power on.

Motion Processing.

Converts the acceleration commands from the host computer to platform position commands in each degree-of-freedom by use of a washout algorithm. Utilizes a dynamic braking algorithm to prevent overstroking any actuator in the presence of large amplitude commands. Performs the transformation to convert the degree-of-freedom position command into six individual actuator position commands. Accepts degree-of-freedom position commands from manual controls in Maintenance Mode.

Design Features.

The use of a high performance microprocessor for motion system control allows incorporation of a number of features that have not been available on previous systems. A design premise was to fully utilize the available computing processing power and to incorporate as many useful functions as possible. These attributes include dynamically variable washout parameters, stored set-up conditions for various vehicle types, and enhanced operational mode controls and indicators.

Variable Washouts.

The ability to change washout parameters between two sets of values during a simulated flight provides a solution for dealing with the transition from cruise flight simulation to ground handling simulation without needing to stop the simulation run and change values. Using different washout characteristics for ground handling simulation than for

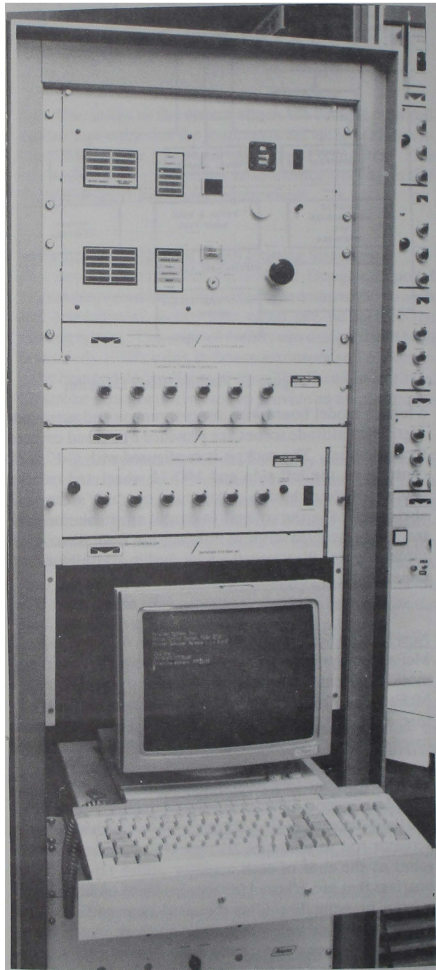


Figure 3. Control Console

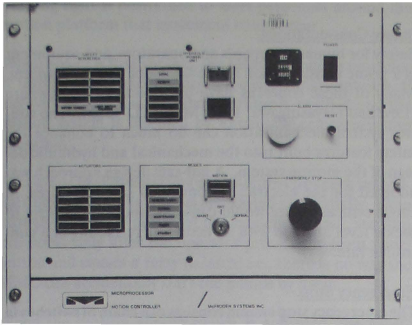


Figure 4. Microprocessor Motion Control panel

cruise flight conditions (Parrish and Martin, 1983) is a desirable feature. Future requirements may dictate other situations where different washout parameters are required. Such an implementation also provides a convenient means for investigation of various washout schemes. The fundamental washout algorithm incorporated into the motion base software uses the coordinated, non-linear adaptive scheme developed at McFadden Systems Inc. and used on several small McFadden motion bases with analog electronic controls. Similar algorithms are in general use (Rolfe and Staples, 1989).

Variable parameters for the second order washout filters include time constant, damping ratio and acceleration scaling gain. For example, during an approach the dynamic parameter change would be initiated by the simulation computer via a discrete signal that is related to a threshold value of altitude. The variable parameters of the washout filters in all six axes are then linearly ramped from the cruise values to the landing values. Switching back to the cruise values is not permitted until the original ramp function is complete. The ramp time is set prior to simulation by the user. Naturally the ramp time must be slow enough not to cause a noticeable disturbance.

Another Douglas Aircraft requirement was to be able to easily accommodate two different types of aircraft without tedious reloading of washout gains and other parameters. This requirement was met by maintaining two sets of washout parameter tables for each of two different aircraft, for a total of four tables. Prior to a simulation run the operator must select the aircraft type or accept as a default the last one used. A typical operating scenario would be C-17 simulation in the morning and MD-11 in the afternoon. The easy change-over ability for washouts and acceleration gains facilitates this mode of operation.

Operational Modes.

The primary operating mode, either Normal, Maintenance or Off, is selected by a keyswitch. The mode switch is located on the MMC front control panel in addition to pump controls, platform motion control, indicator lights and an emergency stop button.

Normal Operation.

Prior to beginning a simulation flight, the hydraulic pumps are started using the MMC front panel pushbutton. After cockpit personnel have boarded the platform and have been secured the motion engage sequence may be initiated by either pilot or co-pilot, from their adjacent remote control panels, or by the control room operator via the front panel switch. To comply with MIL-STD-1558, a motion consent switch is provided to lock out control room start-up without cockpit consent. Pressing the MOTION switch causes the powered stairway to pull back and then the platform slowly rises to the neutral position. The cockpit remote control panel and the cockpit mounted host computer control terminal permit complete authority over the simulation from the cockpit. Motion is disengaged by again pressing the MOTION switch, causing the platform to descend to the egress position and the stairs to be lowered.

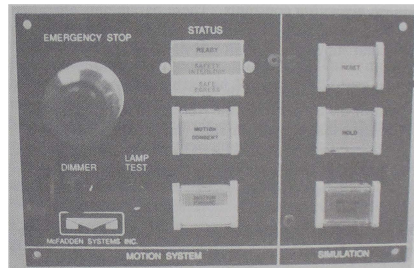


Figure 5. Cockpit Remote Control Panel

Standby.

After the platform reaches neutral position the motion system is in a standby condition. Simulation may be started by pressing the MISSION START button on the cockpit remote panel. Gains on platform motion commands are slowly ramped from zero to their nominal value to avoid abrupt motions. If desired, a motion override command may be entered via the host terminal that will permit non-moving or fixed base run with the platform held stationary at midstroke. Whenever simulation is suspended, either by pressing the HOLD or RESET buttons on the cockpit remote panel, the motion system returns gradually to the level neutral position with motion commands from the host disconnected in a smooth manner to prevent jarring motions.

Maintenance Mode.

When this mode is selected, motion commands from the host are ignored, and the platform may be controlled in each of the six degrees-of-freedom by front panel control knobs. Connectors for external command sources are also provided to aid testing. A frequency response analyzer may be connected to these command sources for evaluation of system performance by either measurement of servo leg position or by measurement of inertial sensors on the platform. Testing by degree-of-freedom rather than by a single actuator provides a realistic appraisal of system response.

User Alterable Parameters.

Various system features and parameter constants may be altered by the user. Software switches and parametric values are stored in non-volatile memory, and are altered via the video display terminal. Numeric values are entered as floating point data. Software switches are entered as either True or False.

Alterable motion drive parameters include six axes of washout parameters and acceleration gains, cross-coupling gains and turbulence filters. Values are stored for two vehicles and two flight conditions for a total of 135 user adjustable motion algorithm constants. These parameters allow the user to tune the motion drive algorithms to precisely suit the particular simulated aircraft and flight conditions.

Adjustable time intervals include:

- Motion engage cycle time (retract to midstroke).
- Motion disengage cycle time (midstroke to retract).
- Transition time from STANDBY to operational gains.
- Transition time for changing between the two washout flight conditions.

Option select software switches are available for the following system features:

- Hydraulic pump selection allows one or more of the three pumps to be disabled. The de-selected pumps will be excluded from the startup sequence.
- Vehicle motion drive parameter selection allows selection of motion algorithm parameters that have been tuned for one of two selected aircraft.
- Maintenance Mode limit switch override allows powered testing of the actuator dashpots.
- Motion algorithm test flags permit testing of the washout and motion drive algorithms with the adaptive features disabled.

Safety Considerations.

Safety for human occupants and operators, and equipment is a prime consideration in the design of a motion system. Many of these concerns have been addressed elsewhere (Levi and Hayashigawa, 1988; MIL-STD-1558). The control system mechanization can do much to enhance the safety features built into the mechanical and hydraulic design of the motion system. A fail-safe design approach was adopted for the electrically controlled safety features: a single critical failure would cause the motion platform to seek a level attitude with all actuators retracting at a controlled rate.

Emergency Stop.

The emergency stop system consists of a chain of switches in series, with each switch assigned to a particular critical failure condition. Each switch in the chain carries a small holding current for the emergency stop relay. The opening of any one of these switches will cause the relay to drop out, which removes power from the pump motors, de-energizes the settle valves causing actuators to retract, and de-energizes the pressure dump valve allowing pressure to bleed from the accumulators.

Switches in the emergency stop chain are sensed by the MMC, causing the microprocessor to take corrective action in addition to emergency stop relay. For example, if an actuator limit switch is contacted, the relay will remove 28 v. power from all energized circuits in addition to the microprocessor using the normal shutdown controls to turn off pumps, etc. By using the sense lines in the emergency stop chain, the MMC will identify the cause of the failure by illumination of the appropriate indicator light. Note that the operation of the emergency stop system is entirely independent of the microprocessor system.

The emergency stop chain elements:

- Actuator limit switches
- Low hydraulic pressure
- Critically high oil temperature
- Critically low oil level
- Emergency Stop button (cockpit, console, pump)
- Facility power failure
- MMC initiated Emergency Stop condition, including:
 - Timeout of CPU watchdog timer
 - Analog power supply failure
 - Excess servo acceleration

Safety Warnings and Indicators.

Non-critical safety related events are sensed by the MMC and the appropriate indicator lamp is lit. The operator is alerted to non-critical items by an intermittent audible tone; critical events are signaled by a continuous tone. Whenever hydraulic pressure is applied to the system a rotating red

warning light is positioned to alert personnel around the motion platform that movement could occur.

When power is first applied, the MMC executes a sequence of tests to verify functionality of the electronic hardware. If any part of this test fails the operator is notified and the motion system is disabled by forcing an emergency stop condition.

Because of the large starting currents drawn by each individual pump motor, the motors are started sequentially. Even though the MMC performs the sequencing task, start-up can occur only after the operator presses the front panel button and causes a relay to latch; i.e., there is no way the MMC can accidentally start the pumps by itself.

When the motion engage switch is activated, the MMC examines the servo valve current for each actuator. If the sign and magnitude are proper to force the actuator to retract, the MMC will begin generation of the servo position command ramp that causes the actuators to extend to midstroke. This safety check prevents any actuator from unexpectedly "jumping" caused by an electronic component failure or misadjustment in the servo controller electronics.

After simulation has begun, accidental opening of the cockpit door or a failure in the host computer data link will force the system to return to the Standby condition. Computer data link failure is signaled to the MMC via a discrete generated by a watchdog timer in the MicroVAX II.

Host Computer Interface.

Platform acceleration and steady state position commands are sent from the host to the MMC as ± 10 v. analog voltages. Discrete events are communicated via relay contact closures. The analog interface was convenient since the existing MicroVAX II was equipped with analog outputs. It also allows simple interfacing to test equipment, such as frequency response analyzers and function generators.

An optional digital interface for the host computer is available using a 64 Kbaud, RS-422 serial interface. This has the advantage of using low cost hardware at the host computer end and simplifying the cabling requirements. A digital function generator that runs on an IBM-PC/Compatible has been developed to provide test signals. Since the serial data interface requires no additional hardware on the MMC, either analog or digital interface is available for selection by the user.

The Microprocessor Solution.

When a microprocessor based system is a replacement for an established analog system, care must be exercised to ensure that the digital system is at least as good as, and

hopefully better than, it's analog predecessor. Many of the features described in preceding paragraphs have been implemented with good success on previous McFadden motion systems using a combination of discrete logic and analog circuits. For a simulator motion system, smoothness and fast response are primary design considerations, and require particular attention to the quantization effects inherent in digital systems.

Modern microprocessors have made digital control for high performance systems a practical and desirable approach for new simulator installations. Benefits offered by a digital motion system include:

- Accurate and consistent algorithms independent of component variations
- Wider range of adjustment of time constants and gains
- Easier implementation of complex control algorithms
- Independence from host simulation computer
- Easier modification of parameters by user
- Lower manufacturing and checkout costs
- Simplified maintenance
- Upgradeable functionality
- Easily documented parameter values

Iteration Rate.

The rate at which the digital control implementation interfaces to the analog signals is a determining factor in both system smoothness and induced time delay. The sampling theorem tells us (Franklin and Powell, 1981) that if the sample frequency is twice the highest frequency of interest, the desired analog waveforms may be produced. This suggests that if the highest important frequencies are about 10 Hz (Rolfe and Staples, 1989) a sample rate of 20 Hz. would be adequate. However, the sampling theorem does not take smoothness into consideration. Other sources (Gebman et.al., 1986) suggest that motion systems using data rates of 30 to 40 Hz. give adequate results.

The effect of sample rate on induced time delay is obvious, however the effect on smoothness depends on the slow rate or maximum frequency component in the output waveform. High slow rates produce large changes in the output between samples.

Resolution.

Like sample rate, the A/D and D/A resolution has an important impact on smoothness. Using 12 bit A/D converters gives an acceleration command resolution of 0.00049 g's with a 1 g scale factor. For output, 12 bit D/A converters give servo position resolution of 0.0176 inches.

Analysis.

The impact of sample rate and resolution on the smoothness of the motion platform was evaluated by computer simulation. A simplified math model of the leg servo was used to determine servo accelerations in response to quantized

position commands. Because of the synergistic action of the six legs an approximate amplification factor of 1.3 was applied to estimate platform accelerations.

Assuming that the output sample rate is fixed at 200 Hz, or .005 sec. per sample, output position commands can only occur at that rate. Servo bandwidth for analysis was 3 Hz. To achieve the maximum velocity of 36 in./sec. requires that each output sample be 0.18 in. Note that this is ten times larger than the resolution capability of the D/A converter. The simulated acceleration transients resulting from a 0.18 inch step position command are equivalent to 0.0101 g's at the platform. These accelerations are small because the servo response is acting like a filter to attenuate the 200 Hz. stair-step signal being applied to maintain the steady state velocity. Note that decreasing the sampling frequency will cause a corresponding increase in acceleration transients. Likewise, using a low response servo provides more filtering action, allowing use of a lower sample rate; but slow servos reduce overall response. If advanced flight simulator motion bases are to utilize fast servos with bandwidths exceeding 1-2 Hz., output data rates over 100 Hz. to the position servos will be required to keep spurious accelerations within acceptable levels.

MIL-STD-1558 allows a maximum acceleration transient of 0.015 g's while moving at a constant velocity. To remain within this limit requires that the sample frequency be approximately 133 Hz. Actual testing of the hardware has shown that 125 Hz. gives very satisfactory results.

Effects of output resolution were investigated in a similar fashion, by applying position steps of .0176 in. at a very slow rate. Analysis showed platform accelerations of 0.0135 g's, indicating that 12 bit D/A converters are more than adequate.

Computation Time.

To be able to achieve a sample rate of 125 Hz. requires that all computations be completed in less than 0.008 sec. Algorithm execution times were estimated by assembly language coding of certain key functions, such as integration, and adding cycle times for all major operations in the algorithm. Comparisons were made between several microprocessors and a 16 Mhz. Motorola 68020 was selected. Estimated motion algorithm execution time was about 0.0045 sec., allowing sufficient margin for program growth.

A Motorola 68881 numeric co-processor was also used to permit the programming convenience of using floating point arithmetic. Even though 16 bit integer arithmetic is nearly four times faster and has sufficient resolution, the difficulty in maintaining proper scaling for good resolution substantially increases the programming effort.

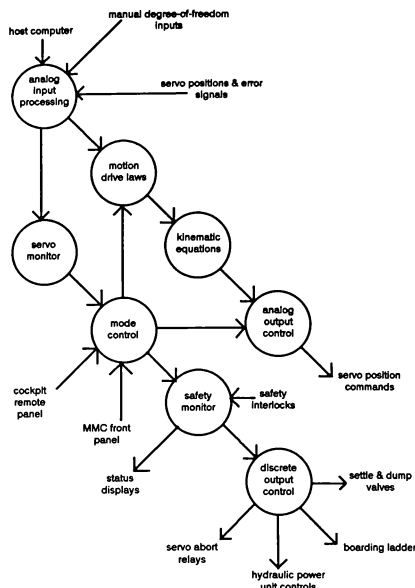


Figure 6. Data Flow Diagram

Software Architecture.

A background/foreground scheme was used as the basis of the software design. The complexity and computational overhead of a real-time operating system was not required for the MMC, specially since all software is stored in EPROM, i.e., no disk operations are used. All foreground functions are executed in a single interrupt scheme. The only other interrupts in the system are for failure conditions. A hardware timer generates interrupts at the sample rate of 125 Hz. Motion algorithm calculations are done in the interrupt routine. Monitoring and operator interface functions are performed as a background function. The foreground function must not use all of the 0.008 sec. frame time so that some time is available to the background functions. The actual system completes interrupt calculations in about 0.0055 sec., and a complete pass through the background functions is completed in about 0.032 sec. Figure 6 shows a data flow diagram for the major system functions and their partitioning between background and foreground. Figure 7 shows the timing relationships between background and foreground functions. Figure 8 shows the continuous background loop executing primarily discrete data processing and I/O. Analog data processing is done during the interrupt routine. The debug monitor is invoked by the non-

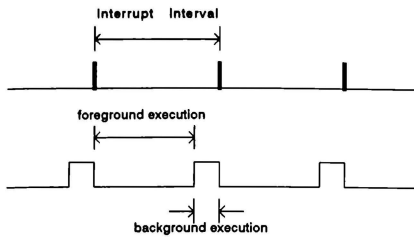


Figure 7. Software Timing

maskable interrupt (a pushbutton) after first safely aborting the motion system control.

Motion Controller Design.

Hardware design is based on the VME bus for several reasons. There are hundreds of vendors for VME bus compatible boards of every description; its popularity insures that there will be support for many years to come, along with reasonably priced hardware. Bus performance is entirely adequate for the motion system application, if some simple precautions, mentioned below, are taken. Figure 9 shows the VME boards installed behind the front panel of the MMC. The CPU board is a Motorola MVME 133-1 single board computer using a 68020 microprocessor and 68881 co-processor. A separate memory board is used for program storage in EPROM and user alterable parameters are stored in battery-backed RAM. Analog and digital I/O boards are from Xycom. The analog input/output boards provide optical isolation between the VME bus and the analog signals.

After a survey of available products, we found no commercial anti-alias filters on a VME board suitable for use in a control system. Since all channels of the high-speed Xycom XVME566 A/D were utilized, a custom 32 channel anti-alias filter board was designed using quadratic filters. Anti-alias filters are essential for ensuring that extraneous noise is not reflected in the control bandwidth via the sampling process.

Programming Techniques.

Nearly all coding was done using the "C" programming language. The popularity enjoyed by "C" for embedded systems programming is well deserved. The source code for the MMC control program is over 150 pages, but the compiled run-time code only occupies about 30 Kbytes of memory. Virtually no "C" library functions were used. All low level hardware drivers were written in "C"; the only sections that required assembly language programming were the CPU initialization sections and the entry routine for interrupt processing.

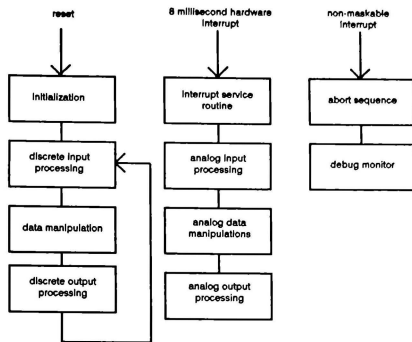


Figure 8. Process Flow

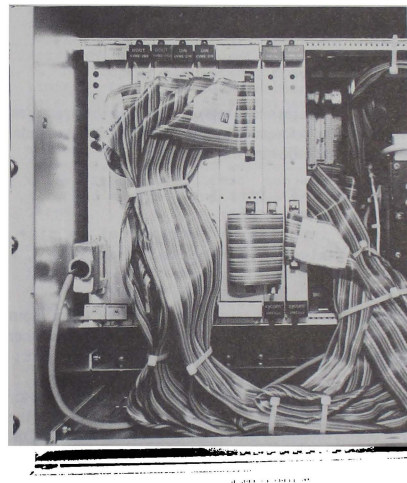


Figure 9. VME card cage

A software engineering goal was to adhere to principals of modern structured programming practice, both to facilitate initial design and debugging and to simplify future program maintenance and modification. Because of the modularization needed for structured programming, numerous "calls" between modules are required. Because "C" passes data between subroutines on the stack, extensive use was made of

pointers to minimize the overhead when making function calls, generally requiring the passing of only one parameter. This method was also very convenient when handling data and algorithms for each of the six degree-of-freedom axes. A generic data template or structure was established for each related type of data, whether runtime variable or constant. Since each of the six axes used nearly identical motion drive algorithms, data could be easily manipulated by changing the pointer to the appropriate axis.

As an example, consider the following "C" declarations for a data structure containing acceleration parameters for each of six axes:

```
struct dof_parm{
    unsigned short accel_channel;
    double accel_scale;
    double accel_gain;
}pitch, roll, ...heave;
```

The elements may be easily referenced by forming an array of pointers to the data structures.

```
struct dof_parm *axis_ptr[6] = {
    (&pitch),
    .
    (&heave) }
```

With data structures and arrays of pointers to the structures, all data may be referenced by pointers when doing calculations, as in the following fragment:

```
for (i=0; i<6; i++) {
    ptr = axis_ptr[i];
    *(accel_input[i]) = (ptr->accel_gain) *
        (analog_in(ptr->accel_chan));
}
```

In the above, the channel number is passed to an input routine, results multiplied by a gain and stored in another data structure referenced by the pointer `accel_input`.

A speed-up technique was used to reduce time required for instruction fetches over the VME bus. At power-up the CPU begins executing the initialization from the EPROM memory board on the bus. It then moves the remaining executable code and constants from EPROM and battery-backed RAM memory to the high speed RAM on the CPU board. When execution begins on the CPU board, bus access is required only when servicing the input/output boards. This technique accounts for a 25% improvement in execution speed over direct bus operation.

Two high speed discrete outputs were assigned to measure the execution time of foreground and background software loops. The execution time and relationship between background and foreground functions are easily observed on an

oscilloscope. This feature will remain in the software to facilitate timing evaluation as software functions are added in the future.

Software Development System.

A low cost development system was used for compiling and debugging the code. An IBM-PC compatible computer was used for the workstation hardware. A Microtec Research 68020 cross compiler and assembler were used to compile code. An Atron "BugBuster" 68020 emulator with overlay memory was used to download and test the code while it was running on the actual hardware. A test panel was used to simulate the many sensors, solenoids and relays on the actual motion base hardware. Provisions for analog signal I/O was also on the test panel. The test panel permitted the software to be nearly fully operational the first time the MMC was connected to the motion base hardware.

Conclusions.

The use of an embedded microprocessor for control of simulator motion platform has been successfully demonstrated. The McDonnell-Douglas simulator described here has been providing high fidelity motion simulations on a daily basis for nearly one year. Though no formal evaluation has yet been conducted, there is general acceptance and praise from pilots.

A high level, structured programming language has been demonstrated to yield results suitable for high performance embedded control. The use of a stand-alone software based control system has made the introduction of new features and more complex motion drive algorithms a relatively painless experience for the user.

References

Six-Degree-of-Freedom Motion System Requirements for Aircrewmember Training Simulators, MIL-STD-1558, 22 February 1974.

Flight Simulation, J.M. Rolfe and K.J. Staples, Cambridge University Press, 1989

"Specification Considerations for a Small Motion Base", R. Levi and L. Hayashigawa, AIAA Flight Simulation Technologies Conference, September, 1988

"Application of Nonlinear Adaptive Motion Washout to Transport Ground-Handling Simulation", R.V. Parrish and D.J. Martin, Jr., NASA Technical Memorandum 84568, 1983.

"Assessing the Benefits and Costs of Motion for C-17 Flight Simulators: Technical Appendixes", J.R. Gebman, et.al., The Rand Corporation, N-2301-AF, June 1986.

Digital Control of Dynamic Systems, G.F. Franklin and J.D. Powell, Addison-Wesley, 1981.

Yorke J. Brown*
Cardullo, Brown and Associates
Binghamton, NY

Frank M. Cardullo*
State University of New York
Binghamton, NY

Grant R. McMillan*
Armstrong Aerospace Medical Research Laboratory
Wright-Patterson AFB, OH

Abstract

Because of the difficulty of providing sustained acceleration in a limited space, flight simulators have historically been unable to faithfully reproduce many important aspects of the pilot's force and motion environment. Although simulator designers have devised numerous methods of cuing acceleration onset, vibration, shock, and control force, the ability to produce useful simulations of the effects of even moderately sustained acceleration is still extremely limited. The problem is compounded by the belief among some that incomplete force and motion simulation is worse than no motion simulation at all. As a result, many training simulators are specified without platform motion systems, and users frequently disable those devices (such as G-seats) which are installed. A number of recent advances in technology and in psychophysiology, however, hold the promise of significant improvements in the ability of simulators to provide meaningful cuing of the force and motion environment, including sustained high-G flight.

This paper reports on a recent study which explored the psychophysiological mechanisms of motion perception and a variety of methods for creating a synthetic aeronautical force and motion environment. Both old and new ideas were investigated in an effort to determine their applicability and feasibility in the light of new technology for implementation. The newer ideas included several novel and highly speculative approaches. Techniques such as Lower Body Negative Pressure (LBNP), Peripheral Vision Occlusion, Thermal Cutaneous Cue Enhancement, Direct Limb, Head, and Equipment Loading, Vibromyesthetic Illusions, and Electroneural and Electromuscular Stimulation were treated.

Introduction

Most of the technology of the modern flight simulator is extremely advanced, allowing unprecedented acceptance of simulation in research, aircraft design and development, and aircrew training. Accurate simulation of the pilot's force

and motion environment, however, still poses a major obstacle to the completeness and fidelity of piloted simulators. The fundamental problem is simply that it is physically impossible to reproduce sustained acceleration in the limited volume available to a simulator. Even the forces associated with sustained aircraft acceleration can only be replicated if the designer is willing to accept spurious counterforces. Modern simulator motion platforms can simulate the higher frequency portions of the force and motion spectrum by providing stimuli which match the onset of acceleration changes, followed by "washout" to a neutral position. This approach has proved useful, particularly in helicopter simulation, but is clearly limited in that it does nothing to simulate the effects of sustained acceleration. Furthermore, if the simulated aircraft is particularly agile, or if the motion cuing algorithm is inappropriately designed, the motion washout may not be fully subliminal, thus subjecting the pilot to spurious and confusing accelerations.

A number of other devices, notably the G-seat, have been developed in order to fill in the lower frequency regime of the force and motion spectrum. Nevertheless, despite isolated successes, no combination of techniques has achieved acceptance as a standard method of full-spectrum motion cuing. In order to address this issue in a systematic way, the US Air Force, through the Armstrong Aerospace Medical Research Laboratory (AAMRL), commissioned a study (3) which explored the psychophysical mechanisms of motion perception and a variety of methods for creating a synthetic aeronautical force and motion environment. The study aimed at consolidating knowledge of existing techniques, and exploring new and speculative techniques for force and motion cuing that might be applicable to the simulator of the future. This paper outlines several of the techniques treated in the study, and discusses aspects of their usefulness and practicality.

The purpose of most of the components of a flight simulator is to provide accurate

* Member, AIAA

and faithful reproduction of the physiological cues a pilot experiences in actual flight, and the goal of the simulation engineer is to develop ways of improving the accuracy and faithfulness of the cues provided. The somatic cues associated with sustained acceleration are particularly difficult to present in a faithful way, however, and it may therefore be necessary to modify somewhat the goal of physiological fidelity.

A faithful reproduction of acceleration-induced forces and body pressures is difficult because any simulation-appropriate applied force or pressure must be balanced by a simulation-inappropriate force in the opposite direction. For example, the force on a pilot's seat is always equal to his weight. In the aircraft, under acceleration, this force increases proportionally to the z component of acceleration--and the sensation of "seat-of-the-pants" pressure is thus an important acceleration cue. In the simulator, where there is no actual acceleration (ignoring onset cues from a motion system), the force on the pilot's seat is constant and equal to his weight. There is no physically possible way of increasing this force except by pushing down on the pilot; pushing up on the seat is impossible without displacing the pilot upwards.

The pilot's perception of his weight on the tissue of his buttocks is due to compressive stress, however, not force. So the simulation engineer has the opportunity to provide an increase in stress as an acceleration cue. The G-seat commonly used in aircraft simulators cues acceleration by reducing the area of contact of the pilot's buttocks with the seat pan thereby increasing the actual stress applied to a localized region (the tissue padding the ischial tuberosities). This action is not a completely faithful simulation of the effects of acceleration, but it does provide some appropriate cuing through natural stimulation of the same physiological receptors as are active in the real world.

Given the difficulty of actually applying forces similar to those encountered in the real world, it is reasonable to ask if other means of stimulating the involved physiological sensors--or of amplifying the effects of available stimuli--are practical. A simple elaboration of the G-seat concept might be to fit the simulator seat with an array of dowels which could be pushed up through the seat cushion to apply high localized pressure to the pilot's buttocks in response to simulated acceleration. The sensation would not be entirely realistic, but it would provide a dramatic--and physiologically relevant--acceleration cue. Several other methods of stimulating the pilot in unrealistic, but appropriate ways are discussed below.

Lower Body Negative Pressure (LBNP)

One of the more profound physiologic effects of +Gz acceleration is the reduction of head-level blood pressure due to pooling of blood in the pilot's lower extremities and due to the increase in pumping energy required to raise blood against its increased apparent weight. Although to some extent the cardiovascular regulatory system can compensate for acceleration stress, the body's reaction time is slow compared to aircraft acceleration onset times, and it is effective only up to a few Gs. The most obvious consequence of the reduction of head-level blood pressure is the "G-dimming" or "acceleration blackout" phenomenon in which collapse of the retinal vasculature results in a progressive narrowing of the pilot's field of view. Prolonged exposure to low head-level blood pressure eventually results in loss of consciousness, a phenomenon known to pilots as G-induced loss of consciousness (GLOC).

Although sustained acceleration is impossible to provide in a simulator, it is possible that the physiological effect of pooling blood in the lower body may be stimulated through the reduction of atmospheric pressure over the lower body. In terms of its cardiovascular effects at least, Lower Body Negative Pressure (LBNP), as the technique is called, has been suggested as an analog for vertical acceleration. LBNP has been studied for various medical reasons since the mid-nineteenth century, and, since the 1960s, has served as an analog for terrestrial gravity in preventing the cardiovascular deconditioning experienced by astronauts exposed to microgravity for long periods. That LBNP could serve as an analog for aircraft acceleration in a flight simulator was first proposed by Howard (5), while various investigators (eg, Latogola and Trent, (13)) have explored the idea since. Nevertheless, no truly comprehensive study has been attempted which would establish an effective method of using LBNP to simulate aircraft accelerations.

In spite of the appealing parallels between the abilities of LBNP and Gz exposure to produce pooling of blood in the lower extremities and to produce unconsciousness, their physiologic mechanisms are not entirely similar. Vertical acceleration acts on the cardiovascular system mainly by increasing the amount of work the heart must do to raise blood out of the lower body against its increased apparent weight, whereas LBNP acts by merely expanding the compliant venous vasculature of the lower extremities. Although there is considerable evidence that pilots experience similar sensations when exposed to LBNP and when exposed to Gz, there is no data supporting any quantitative correlation between LBNP and Gz in terms of levels of exposure or onset

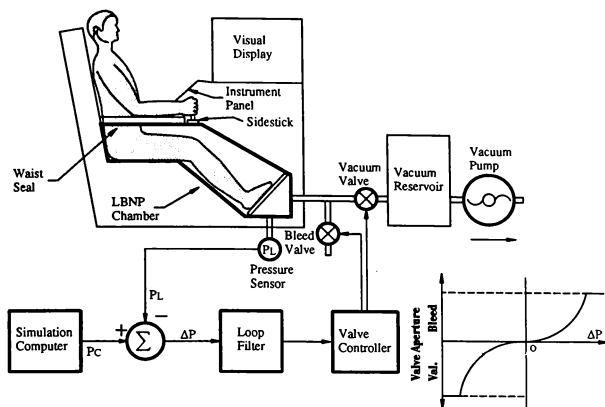


Figure 1. Proposed experimental device for the study of LBNP as a high-G simulation technique. The pilot's lower body is contained in a sealed box, the pressure of which is varied as a function of simulated Gz. Only a modest vacuum pump is required since the reservoir allows for rapid pressure reductions. Pressure increases are handled by a bleed valve.

rates. Krock (8) argues that no similarity should even be expected (except, perhaps, by matching very slow Gz onset to very rapid LBNP onset), and is currently performing experiments to test his assertions.

Since acceleration exposure is such an important part of the tactical pilot's working environment, awareness of its effects and training in protective measures are vital both to the safety of the pilot and to his mission effectiveness. Although LBNP does not appear today as promising as it once did, it still deserves attention as an element in Gz training devices and in air combat simulators. Figure 1 shows an experimental LBNP device proposed to study the applicability of the technique to simulation of the high-Gz environment.

Peripheral Vision Occultation

When a pilot experiences a reduction in head-level blood pressure as a result of sustained acceleration, the retinal vasculature collapses under the intra-ocular pressure (7). The collapse begins in the periphery of the retina and, as the blood pressure declines, progresses inward. Regions of the retina thus deprived of nourishment become insensitive to light and the pilot experiences a progressive narrowing of his field of view known as G-dimming or blackout. Initially, the pilot loses only his peripheral vision, but as the acceleration increases, the

clear field becomes progressively smaller until the entire visual field is lost. If the eye-level blood pressure is restored, whether as a result of cardio-vascular compensation, straining, the action of a G-suit, or relaxation of the acceleration itself, the pilot's vision returns. G-dimming is a particularly vivid acceleration cue, and is one which pilots rely on for gauging their physiologic state under acceleration. It would therefore be very advantageous to provide this cue in flight simulators used in training for high-G flight.

If the approach of actually reducing the pilot's eye-level blood pressure in order to produce G-dimming is rejected, it is still possible to simulate the purely visual effect by occluding the pilot's peripheral field with an appropriately designed variable transmission visor which could act as a function of simulated acceleration. A variable transmission visor system would consist of three main components: the visor itself, an oculometer for determining the instantaneous position of the eye, and a computational system for interpreting the oculometer, computing the appropriate level of G-dimming, and driving the visor. A general schematic of a proposed arrangement is shown in figure 2. Although the visor is fixed to the pilot's head, the oculometer is required because the occulting pattern provided by the visor must be centered on the eye. The occulting mask provided by the visor must not only change size and

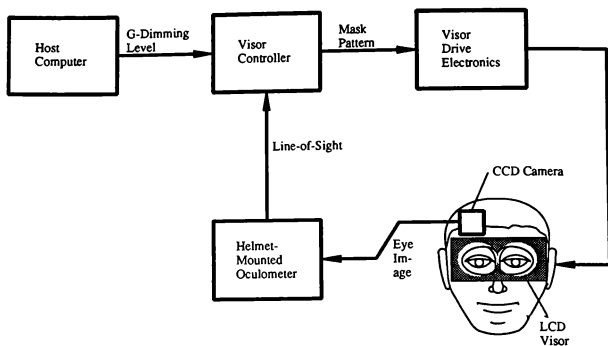


Figure 2. Proposed configuration for an LCD variable transmission visor for simulating G-dimming. The size of the clear area of the LCD panel is controlled by the host computer as a function of Gz. The clear area is positioned on the panel so as to remain centered on the pilot's line of sight. The line of sight is determined by a helmet mounted oculometer. The LCD panel itself is a time-multiplexed pixel array using thin film transistor technology to maximize contrast.

shape as a function of acceleration, it must track the motions of the eye. The visor, therefore, must be capable of writing an arbitrarily shaped opaque area anywhere on its surface and must be capable of rewriting its entire surface during the period of saccadic suppression--about 60 milliseconds.

Kron, Cardullo and Young (9) have suggested that a wide aperture, transmission mode, matrix addressable liquid crystal device (LCD) is a possible choice for the visor. Their exploration of this idea follows a patent by Hoyt, et al. (6) on an LCD visor which uses masks fixed on the surface, a technique which is clearly feasible but which does not allow for pilot eye movements. Although some difficulties remain, advances in LCD technology have now made possible a variable transparency visor with individual matrix addressable pixels providing full freedom in the shape and placement of the occulting mask.

There are two remaining significant problems with the LCD visor concept. First is that a matrix addressable array of the required size will exhibit fairly low contrast because the dwell time on each pixel is a small fraction of the entire matrix scan time. This problem has potential solution in the new generation of LCDs being developed for flat-screen television applications. These devices include an array of thin-film transistors which are integrated directly into the matrix and which hold the bias voltage on each pixel individually until the next scan. The second problem is that, even in

the clear state, LCDs polarize the light they transmit. Consequently the maximum transmittance of the visor is less than 50 percent. Wearing the visor will be rather like wearing sunglasses--a troubling prospect given the brightness of even the best simulator visual displays. Nevertheless, for many training tasks this drawback may not be a critical.

Thermal Cutaneous Stimulation

A willingness to depart from a requirement for absolute realism opens even more avenues for useful cuing. There is some evidence, both experiential and physiological, that the mechanisms of pressure sensation and temperature sensation are related. It is well established that there is not a simple mapping between cutaneous receptor type and sensitivity of particular body regions to mechanical stimulation (19). Further it is known that the cutaneous pressure receptors are also sensitive to temperature (19, 20, 21). It is possible, therefore, that the pressure cue of a G-seat could be enhanced or amplified by the simultaneous application of a temperature change to the pilot's seat. The pilot would not, of course, confuse a temperature change with a pressure change, but the thermal effect may enhance the existing pressure cues, and, realistic or not, may serve as a suitable analog for sustained G-induced seat pressure.

A thermal cutaneous stimulator employing inexpensive Peltier Effect heat pumps would be remarkably simple to implement if the technique were to prove effective. An

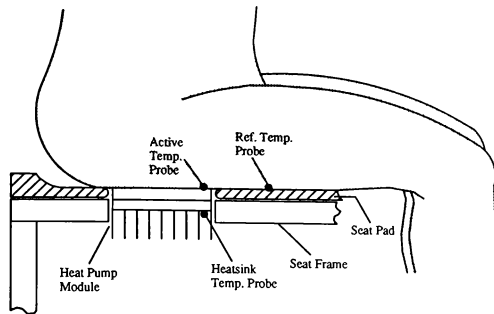


Figure 3. Proposed experimental arrangement for studying the use of thermal cutaneous cuing. A Peltier Effect heat pump is mounted under the seat pan of a G-seat so that it can either heat or cool the pilot's buttocks tissues. Several thermistor probes are provided to monitor the thermal dynamics of the system and allow for precise control of temperature and heat flow.

experimental investigation of the usefulness of thermal cutaneous stimulation must start with an accurate and quantitative characterization of the applied stimulus, and proceed to investigation of the perceptions associated with heating or cooling the pilot's seat in conjunction with other stimuli. Figure 3 shows a proposed Thermal Cutaneous cuing device suitable for integration with a G-seat.

Direct Limb, Head, and Equipment Loading

Although the forces associated with pilot G-loading cannot be applied in a stationary environment without inappropriate counter forces, the constraints on action produced by limb, head and equipment G-forces are major components of G-stress. It seems reasonable that direct application of these kinds of forces would enhance the effectiveness of a high-G simulation. The use of electric or pneumatic actuators for applying forces to the pilot has been explored with some success in the past (1) and deserves further attention. Indeed, direct loading may be the most convincing and useful of all the approaches studied, despite obvious concerns of safety and of pilot acceptance of the encumbrance of various connections to body and equipment.

As a means of avoiding the problem of the need to connect the pilot to cables or arms, the study treated the possibility of using motors carried directly on the pilot's body and integrated into the flight suit. This approach requires lightweight, compact and efficient motors of advanced design, since they must be unobtrusive and yet produce high stalled

torque without overheating. This combination of requirements is still not generally available, but recent developments in magnet technology are beginning to bring this approach closer to practical reality. A forearm loader, such as that shown in figure 4, may well be practical if it is used at a sufficiently low duty cycle.

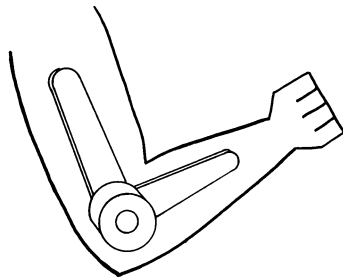


Figure 4. A possible arrangement for an electric forearm loader. A samarium-cobalt pancake motor is carried on the pilot's elbow by plastic arms sewed into the flight suit. The motor, under the control of the host, delivers torque tending to extend the elbow joint.

Vibromyesthetic Stimulation

Lackner and coworkers (10, 12) have discovered that vibration in the range of 100 Hz applied to muscle spindles can create the illusion that the involved limb is in motion. Accompanying this vibromyesthetic illusion is a nystagmus that is compensatory in direction, and which results in an illusory movement of a target light against a dark background. Using ordinary physical therapy vibrators operating at 120 Hz, Lackner and coworkers have demonstrated that the vibromyesthetic effect can evoke illusions of displacement and continuous motion in virtually any direction. Furthermore, Lackner and Graybiel (11) have found that vibration applied to the head can stimulate the semicircular canals and possibly the otoliths.

Although the vibromyesthetic illusions are compelling, they are easily destroyed by contrary sensory evidence. For example, the illusion of forearm flexure created by applying a vibrator to the insertion of the biceps is destroyed immediately by either looking at the arm, or contracting the biceps even slightly. Most of the experiments so far have been conducted either in the dark or with the subjects' eyes closed. It is possible, however, that the illusions may be exploited as part of a simulator motion cuing system if they are used to amplify motion perceptions suggested by other means. Although highly speculative at this point, the vibromyesthetic effect holds promise as a fruitful area of research.

Electroneural and Electromuscular Stimulation

A similarly speculative, but currently more highly developed, area of artificial somatic stimulation is the use of electrical stimuli to induce a sensation of motion or to cause actual contraction of muscles. Wild-eyed simulator designers have long wished to simply install BNC connectors in the backs of all pilots' heads, thereby providing the means to stimulate the brain directly without the necessity for cumbersome cuing devices. Electrical stimulation of the vestibular system, using surface electrodes, has been explored in a number of clinical and experimental studies. Application of small direct current stimuli (0.4-1 mA) reliably produces head and body tilts toward the anode, presumably via otolith mechanisms (22, 24). Application of sinusoidal alternating currents (0.1-2 mA) has been shown to modify the direction, amplitude, and frequency of normal postural sway. Such stimuli can even cause people to fall (4). The work of Dzendolet and his colleagues is, perhaps, the most relevant for simulator applications. For example, Berthold and Dzendolet (2) found that, by using low frequency (0.03-4 Hz) sinusoidal electrical stimulation applied bilaterally to a subject's mastoid proc-

esses, they could arouse sensations of rotational oscillations. The subjects were blindfolded, seated and wore acoustical ear plugs. Three types of stimulation were presented to the subjects in this experiment: mechanical rotation of the chair alone, electrical stimulation alone, and simultaneous mechanical and electrical stimulation. The subjects reported similar sensations with the electrical as with the mechanical stimuli. When mechanical and electrical stimuli were applied simultaneously, there was an apparent interaction manifested by a phase difference between the actual (mechanical) and perceived (mechanical plus electrical) stimuli. This study is especially interesting in that semicircular canal mechanisms appear to be involved.

Two possible implementations of this technology can be imagined: one would use the electrical stimulus alone to provide vestibular responses consistent with simulated aircraft dynamics; another would employ it in conjunction with a motion platform or a dynamic seat as a cue potentiator or to sustain cues after the platform or seat has reached its limit. It would seem, from the interaction described by Berthold and Dzendolet (2), that the second alternative might prove to be problematic. However, more research is required to either substantiate or refute these results, and to gain better understanding of the involved mechanisms. While the electroneural stimulation approach is quite promising, many questions remain: Can the sensations be calibrated? What voltage or current levels are appropriate? How long can the sensations be sustained? Do these artificially-produced sensations have the same effects on pilot behavior as normal sensations?

During the past decade, researchers have made dramatic progress in developing methods of electrical stimulation to be used in prosthetic devices for paraplegics. Petrofsky and coworkers (16, 18) have succeeded in using electromuscular stimulation to enable some paraplegic patients to artificially control their limbs well enough to walk. Electromuscular stimulation might be applied in flight simulators as a means of constraining a pilot's limb movement in accordance with the simulated aircraft acceleration profile. In addition to ambulation, this technology has been applied to hand control in quadriplegics. Lieberman, et al. (14) first demonstrated that electrical stimulation applied to the quadriceps muscles enabled a paraplegic patient to stand. Petrofsky, Phillips, and Heaton (17) describe the control system they developed for walking. Much can be learned from this work in terms of controlling an electromuscular stimulator for simulator applications. According to Petrofsky (15), the key to the implementation is finding a waveform for the stimulus which will generate sufficient force

with the target muscle without injuring the subject, or producing pain. Although pain is not an issue with paraplegics, there is much controversy concerning the possibility of eliminating it in normal subjects.

Conclusion

Advances in technology and in our understanding of the psychophysics of motion perception clearly indicate the possibility of advances in simulator motion cuing which could rival the progress made in visual systems and mathematical modeling techniques over the past two decades. Simulators need better force and motion cuing. With continued research, the simulator of the future can be equipped with devices which create a synthetic motion environment providing stimuli in the critical low-frequency regime, and which simulate the stresses and constraints on pilot action which are central to the pilot's experience of high-G maneuvering. The simulator of the future will not simply be an extrapolation of current technology, or simply an improvement in our ability to replicate the stimuli giving rise to the sensations of flight. Rather, it will involve techniques which synergistically create a synthetic motion environment, perceptually similar to the pilot's actual motion environment.

Although some of the techniques described in the AAMRL study are highly speculative psychophysically, some of them appear to be practical with current technology. LBNP, for example, since it creates such dramatic physiological effects, should be investigated specifically as a G-cuing device, and either ruled in or out as a candidate simulation technology. Thermal cutaneous cuing, on the other hand, is technologically simple, but still requires extensive investigation to assess its applicability and then to develop a protocol for its use.

The currently practical technologies, such as the occulting visor or cable-coupled body loaders could be readily developed for widespread use in the near future, since it is already clear that they are appropriate cuing techniques.

The AAMRL study indicates that technology has placed us on the threshold of a new generation of motion cuing devices. If progress continues, the simulator of the future will perform much better than current devices in the high-G and sustained-G regions of the flight envelope. Perhaps more important, though, is that the force and motion environment of more routine flight maneuvers will also be synthesized more effectively.

Acknowledgment

We are grateful to Gary Riccio for his contributions to the thermal stimulation study.

References

1. BR Ashworth, BT McKissick, "The effect of helmet-loader G-cuing on pilot's simulator performance," The American Institute of Aeronautics and Astronautics, AIAA Paper 78-1573 (1978).
2. HC Berthold, E Dzendolet, "Sensed movement to sinusoidal angular and electrical stimulation," *Perceptual and Motor Skills* 36, 23 (1973).
3. YJ Brown, FM Cardullo, JB Sinacori, GE Riccio, GR McMillan, "New approaches to motion cuing in flight simulators," AAMRL Technical Report, To be published (1990).
4. E Dzendolet, "Sinusoidal electrical stimulation of the human vestibular apparatus," *Perceptual and Motor Skills* 17, 171 (1963).
5. JC Howard, "Feasibility of using lower body negative pressure (LBNP) to simulate transient and sustained acceleration forces," NASA-Ames LTI:239-3 (1976).
6. Hoyt, et al, US Patent No. 3,942,270.
7. EA Jaeger, RJ Severs, SD Weeks, TD Duane, "Visual field changes during positive acceleration," *Aerospace Medicine* 10/64, 969 (1964).
8. LP Krock, Personal correspondence, USAF School of Aerospace Medicine, Brooks AFB, TX (1989).
9. GJ Kron, FM Cardullo, LR Young, "Study and design of high-g augmentation devices for flight simulators," Air Force Human Resources Laboratory, Report F33615-77-C-0055 (1980).
10. JR Lackner, MS Levine, "Changes in apparent body orientation and sensory localization induced by vibration of postural muscles: vibratory myesthetic illusions," *Aviat. Space and Environ. Med.* 50, 346 (1979).
11. JR Lackner, A Graybiel, "Elicitation of vestibular side effects by regional vibration of the head," *Aerospace Medicine* 45, 1267 (1974).
12. JR Lackner, AB Taublieb, "Influence of vision on vibration induced illusions of limb movement," *Experimental Neurology* 85, 97 (1984).
13. MT Lategola, CT Trent, "Lower body negative pressure box for +Gz simulation in the upright seated position," *Aviat. Space and Environ. Med.* 50, 1182 (1979).
14. WT Lieberman, HJ Homequest, D Scott, M Dow, "Functional Electrotherapy: Stimulation of the peroneal nerve synchronized with the swing phase of the gait in hemiplegic patients," *Arch. Phys. Med. and Rehabil.* 42, 101 (1961).

15. JS Petrofsky, Personal correspondence, University of California, Irvine, CA (1990).
16. JS Petrofsky, "Cyberkinetic assisted walking," Proceedings of the IEEE 1986 National Aerospace and Electronics Conference (pp. 759-767), Dayton, OH, May 19-23 (1986).
17. JS Petrofsky, CA Phillips, HH Heaton, "Feedback control systems for walking in man," *Com. Biol. Med.* 14, 135 (1984).
18. JS Petrofsky, J Smith, "Computer aided rehabilitation," *Aviat. Space and Environ. Med.* 59, 670 (1988).
19. TA Ryan, "Interrelations of the sensory systems in perception," *Psychological Bulletin* 37, 659 (1940).
20. JC Stevens, "Thermal intensification of the touch process: further extensions of the Weber phenomenon," *Sensory Processes* 3, 240 (1979).
21. JC Stevens, BG Green, "Temperature-touch interaction: Weber's Phenomenon revisited," *Sensory Processes* 2, 206 (1978).
22. T Tokita, H Miyata, Y Ito, K Takagi, "Diagnosis of otolith and semicircular-canal lesions by galvanic nystagmus and spinal reflexes," *The Vestibular System: Neurophysiologic and Clinical Research*, MD Graham and JL Kemink (Eds.), Raven Press, New York (1987).
23. S Tung, "The integration of punctiform cold and pressure," *Am. J. Psych* 32, 421 (1921).
24. Y Watanabe, H Ino, H Ohi, H Ohashi, H Kobayashi, K Mizukoshi, "Clinical evaluation of vestibular-somatosensory interactions using galvanic body sway test," *The Vestibular System: Neurophysiologic and Clinical Research*, MD Graham and JL Kemink (Eds.), Raven Press, New York (1987).

SIMULATION NETWORKING

Amnon Katz, Ph.D.*
McDonnell Douglas Helicopter Company
Mesa, Arizona

ABSTRACT

This paper describes the methods implemented in the simulation facility of McDonnell Douglas Helicopter Company (MDHC) for local and long-haul networking. The real time system consists of distributed multi processor VME chassis and other components driving four dome simulators and several auxiliary stations. All can be run together interactively, separately, or in various groups. The system is controlled by one or more System Control Stations (SCS) that themselves are distributed systems. Two distinct networks are used locally: an Ethernet for loading, starting, control, and data collection, and a Proteon Pronet 80 for real time communications. One of the principles of the system architecture is separation of the control and command functions and making them invisible to the real time application. This is achieved by an ethernet service called the Stealth Protocol that can read and write remote memory transparently. Long-haul networking with remote facilities is accomplished by a Gateway Computer with access to the local networks and to a standard or special telephone service.

I INTRODUCTION

The MDHC engineering simulation facility, supporting both Apache and LHX activities, is being continuously upgraded. One of the thrusts of the 1989 work was to implement a distributed architecture based on multiple 68020 microprocessors in multiple VME chassis communicating over local area networks.

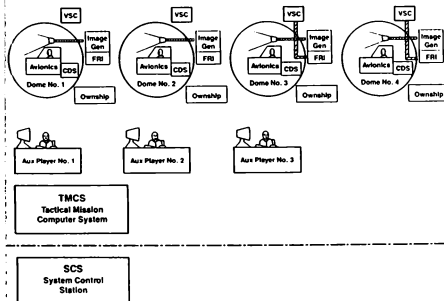


Figure 1: MDHC Distributed Simulation System

The system (Figure 1) supports four dome simulators, three manned auxiliary stations, and Tactical Mission Computers generating an assortment of additional players. All of these can be flown interactively or broken into separate missions.

* Manager, Advanced Development Office, Combat Simulation and System Integration, member AIAA.

The simulation is controlled by one or more System Control Stations (SCS). Each SCS is itself a distributed system including VME chassis, IRIS computers and other components. The SCS is the point from which the operator, the test director, and the analyst configure and control the simulation. This is also the point at which data to be extracted from the simulation is identified and collected.

II LOCAL NETWORKS

The need for networking this diverse system is obvious. Point to point communications would require multiple interface boards in each chassis, quickly outturning the number of available slots.

What is actually used is not one local area network, but two. An Ethernet and a Proteon Pronet 80 (ref 1). The two networks are used for different purposes and are managed differently. The Pronet is used for real time communications only. The Ethernet serves all other purposes including downloading of code, debugging, monitoring and control of the system, and data collection.

In the case of the Ethernet several protocols were developed, all above the TCP/IP level. This approach was adopted to assure compatibility with the diverse computers that are used for development of code and data and from which the diskless targets are loaded. The many levels of software involved create a bottleneck at each node. However, as explained below, this is in a way beneficial.

One undesirable effect of the TCP/IP approach proved to be dependence on the software of Ethernet equipment vendors. They could and did introduce changes into their software products unilaterally. Said changes, which at times were not quite error free, affected the work at MDHC.

The 10 million bits per second Ethernet would not have been equal to the task of real time communications anyway. The collision prone TCP/IP protocols make it even less so: When two nodes attempt to use the net at the same time, both fail and must try again. This wastes network time. The time at which a transaction can be completed is unpredictable and cannot be guaranteed within a given simulation frame. The upshot is that the only a fraction of the physical network bandwidth is usable.

Things are quite different with the Pronet 80. Although physically a star, it is logically a ring, and is operated as a token ring with a single token. A node cannot transmit unless it has the token, and then is limited to a two thousand byte (max) packet. Only one node transmits at a time, and nearly the whole 80 million bits per second bandwidth can be used. True, a node may have to wait for the token. The length of the wait depends on its way intend to transmit. The worst case delay in the MDHC system is close to seven milliseconds. However, this worst case cannot apply to more than one node - the cause of the delay is that other nodes are talking on the net. What's more, once this worst case delay is over, not only the node affected by it, but every node on the net has had its say. The overall net time per frame is predictable, and

communications can be guaranteed to complete within a frame.

The Pronet 80 features broadcast capability - to all nodes or selected groups. This enhances its usefulness as the vehicle for real time communications.

Since the Pronet 80 links only the real time target nodes, compatibility with diverse computers was not a consideration. The Pronet driver was coded in assembly language and designed to optimize the network use. This approach proved straightforward and robust and assured MDHC control over the software.

During integration of the system it was found that fast Pronet communications tended to run into VME bus bottlenecks on the destination node. The design was later modified to be less jealous of network time and more tolerant of delays at a node.

III ETHERNET PROTOCOLS

Four Ethernet protocols were implemented:

1. Code download. This service is required to even start the target computers. Code is cross compiled on a development computer (normally a VAX cluster) and downloaded to the diskless targets in the form of S-Records.
2. File transfer. The target computers use this protocol to read their files of initial conditions and mission specific data.
3. Debugging. This protocols enables the Source level debugger that comes with the Systems Desiners Ada to reach the targets from the development computers.
4. Stealth Protocol. This is by far the most original, versatile and interesting among the MDHC Ethernet protocols. The stealth protocol affords an open window into the memory space of the VME target computers. It makes it possible to "peek" and "poke" in the memory of the target computers from remote Ethernet nodes while the resident code runs undisturbed.

Applications of the stealth protocol include:

1. Setting the mode (INITIALIZE, FREEZE, RUN...) of many simulators from a single point.
2. Reset of remote processors.
3. Specifying initialization parameters and files.
4. Synchronization of Mission time clocks for different processors (ref 2).
5. Collecting data from the simulator system for analysis and study.
6. Reading variables from the simulator program for the purpose of debugging.
7. Writing simulator variables for the purpose of moving players rapidly to a new location ("hyperspace").
8. Writing simulator variables for the purpose of inserting faults and failures.
9. Interfacing systems the design of which did not provide for an interface (see section V).

The Ethernet service is provided in the target chassis by a set of two boards: an Ethernet interface board and a dedicated processor board known as the Command and Control (C&C) board. The C&C board hosts the local component of the Ethernet service which interacts with other boards over the VME bus.

The case of the Stealth protocol may serve as an illustration. The "Stealth Protocol" software, which

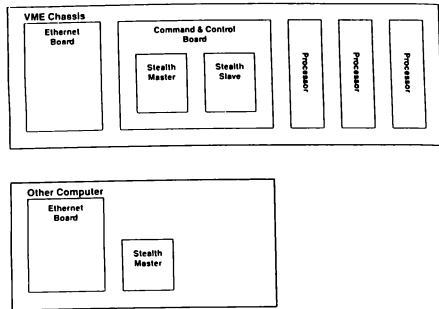


Figure 2: Ethernet Support Hardware.

resides in the C&C board above the TCP/IP level (Figure 2) consists of two parts: The Master which is ready to serve the local applications should they decide to initiate a Stealth Protocol transaction, and the Slave which is responsible for the collection and updating of local data as specified by a remote Master. Non VME systems that can serve as control points - such as VAX and IRIS computers - run the Master part only.

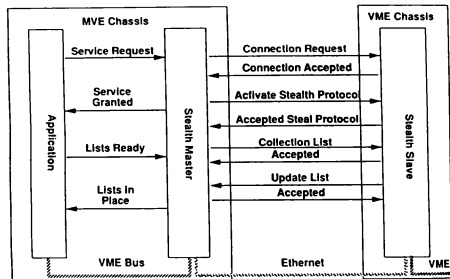


Figure 3: Initialization of Stealth Protocol.

In the following we refer to the application initiating the Stealth Protocol transaction as the control function. In most cases the control function resides in a System Control Station (SCS), which is the Central Control Point for one or more simulators operating together.

The control function initiates a stealth protocol connection by triggering a mailbox interrupt. In the mailbox the control function provides the Stealth Protocol local Master a pointer to a Device Control Block (DCB). The initial contact is made using a common mailbox available to all applications. The Master then assigns the control function a private mailbox for further communications. The control function may also assign the Master a mailbox for acknowledgements and status information. The control function may instead elect to monitor the status information in the control block.

The DCB contains the internet address of a remote

chassis, and pointers to a "read list" and a "write list" (VME addresses of variables to be read and variables to be written in the remote chassis). Fifteen levels of indirection are supported.

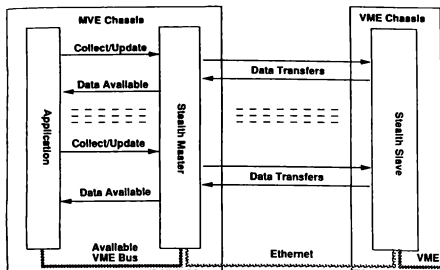


Figure 4: Peeking and Poking by Stealth Protocol.

The Master establishes communications with a remote slave and passes it the read and write lists (Figure 3). The Master then goes into suspension. Upon further interrupt from the control function (Figure 4), the Master alerts the remote Slave, and passes it the list of values to write. The slave writes the values received into the locations in the write list, then reads the locations in the read list and sends the values obtained to the Master. Both write and read are included in each transaction. If only one is desired, the list for the other can be set to NULL.

The control function can issue a CLOSE command. Having received it, the Stealth Protocol local Master closes the sockets assigned for the connection and makes them available for other control functions.

A single Master can maintain open sockets for collecting and updating data from up to 32 different targets at the same time. To change the read/write lists, the control function must issue a close command and re-open with a new list. Any errors detected during transmission are reported to the control function in the "RETURNED STATUS" location of the Device Control Block.

IV SEPARATION OF CONTROL FUNCTIONS

One of the principles underlying the MDHC hardware/software architecture is separation of control and data collection functions from the real time application. The simulation is developed to represent a real world process. Once complete, the software should not be subject to change because of varying requirements of data collection. Rather, whatever data is required is obtained transparent to the real time process by the Stealth Protocol.

Armed with a "data dictionary" constructed from files produced when the real time application is compiled and built, the analyst at the SCS can satisfy his data needs. The analyst may define and redefine his data collection requirements on the fly, and the real time application need never be changed or recompiled because of it.

The principle of separation of real time processes from control and data collection functions is carried a step further by assigning them separate communication networks. The separation of communications networks assures that data collection

traffic cannot interfere with the workings of the simulator. An overambitious analyst could choke his own network and fail to obtain everything he specified. But the basic real time simulation continues to run.

The 10 million bit per second Ethernet would be at best marginal as the real time communications link. But when used as a data collection vehicle with varying demand, the limited bandwidth is somewhat of a blessing. Data collection competes with basic simulation operations for use of the VME bus in each chassis. However, with a bandwidth of 25 million bits per second, the VME bus is not likely to be overwhelmed by the 10 million bits per second Ethernet. The bottlenecks created at each node by the high level TCP/IP protocols provide an added safeguard against the collection process interfering with any real time computer.

V LONG-HAUL NETWORKING

The MDHC facility is developing methods for networking its simulators to other facilities. This should permit to fly high fidelity simulators located in separate facilities together. The potential benefits both for engineering development and evaluation and for training are significant.

When local simulators are networked at MDHC, it is one of the Tactical Mission Computer (TMCS) that serves as the focal point. The TMCS maintains a data base of players, computes weapons flyouts, takes care of meteorological conditions and monitor line of sight between players ... From the point of view of the TMCS players can be internal or external. Internal players are computer generated with their logic and dynamics processed within the TMCS. External players are those simulated elsewhere. These include dome simulators, auxiliary players flown manually from desk top stations or rudimentary cockpits, and computer generated players simulated outside the TMCS.

The TMCS communicates with external players over Pronet 80. It receives position and other data from external players individually. It broadcasts the data of all players, so that any node can be informed of all players taking part in the scenario.

Long haul networking is achieved by a Gateway Computer (GC). The GC is itself a VME chassis with several boards. It communicates with the TMCS over pronet pretending to be one or more external players (that is external to the TMCS). On the far side the GC communicates with the remote player using either a

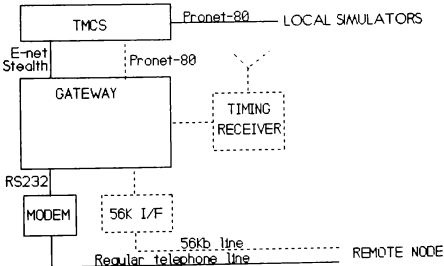


Figure 5: Hardware Arrangement for Long Haul Networking

regular telephone line or a special high speed data line.

The general arrangement for long haul networking is shown in Figure 5. The figure shows both Pronet 80 and Ethernet as possible connections between the Gateway and the TMCS. Ethernet here means the Stealth Protocol used in real time to peek and poke in the TMCS's Players Database. This is a quick and painless way for prototyping. The Stealth Protocol is fast enough to perform this real time task at 60 Hz so long as no significant data collection traffic is present on the Ethernet.

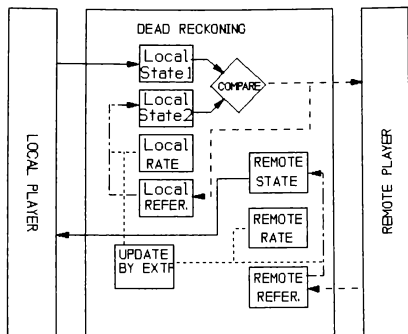


Figure 6: Data Flow for Long Haul Networking.

Figure 5 shows also a timing receiver. This is for the purpose of generating absolute time stamps to be included in the data packets sent to the remote node (ref 3). The flow of data through the gateway is shown in Figure 6. The heavy lines show local traffic occurring every simulation frame (60 Hz). The dashed lines denote communications taking place occasionally and asynchronously. The data received from the remote node is maintained as a Remote Reference. This is used as the basis for computing the current remote state by extrapolation. It is the extrapolated state that is sent to the TMCS every frame. The extrapolation is from the time stamp of the reference to the current time. Hence the need for absolute time stamps.

The time stamp is the elapsed time since the beginning of the current hour (ref 3). This convention reduces the length of the time stamp and also eliminates any confusion due to the fact that the remote player may be in a different time zone.

Note also that the time stamp does not refer to the time at which the data packet was computed, collected, or transmitted. Rather it is the time for which the information in the packet is valid. The timestamp is a Dynamic Time in the terminology of reference 2.

The gateway screens the time stamps it receives to determine whether they can be trusted. Should they fail this test, they are subjected to further checking to determine whether they can be used as relative time stamps. Relative time stamps need only measure the time difference between two adjacent data packets with usable accuracy. They can be used to compensate for the variations in transmission delay, while using a

nominal postulated value for the average delay.

On the local side, the player state (Local1) is updated synchronously every simulation frame. It is shipped to the remote only occasionally. This is done either at a predetermined reduced rate or on an "as needed" basis. In the latter case (Figure 6), a local reference is maintained and a local state "Local2" projected from the reference. This represents the extrapolation being performed at the remote site. When Local2 diverges from Local1 beyond a predetermined tolerance, an update is indicated.

This method of event driven transmissions was introduced by the SIMNET project and termed by them "dead reckoning". The purpose is to reduce remote traffic and the theory is that the statistics of a very large system prevents high activity periods of all nodes from occurring simultaneously. In small systems it may be more prudent to transmit at a predetermined rate, providing for the worst case. This has the added merit of eliminating the need for identical extrapolations on both ends.

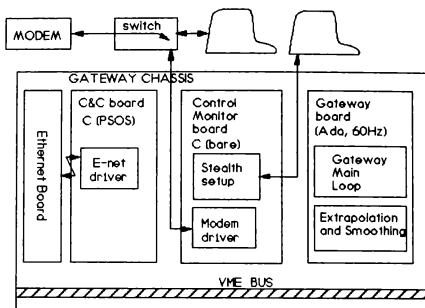


Figure 7: Prototype Gateway.

Figure 7 shows a prototype gateway arrangement that was used recently to network the McDonnell Douglas high fidelity Apache helicopter simulator to aremote site. The McDonnell Douglas simulator was in Mesa, Arizona. The remote player was in Fort Worth, Texas, nine hundred miles and one time zone away. The remote player was a high fidelity Bell 222 simulator located in the Bell Helicopter plant in Fort Worth. This proof of concept demonstration was run over a standard telephone line with a 2400 baud modem. Remote communications were at 2 Hz with local smoothing. This was shown to a group of attendees of the 11th Interservice/Industry Training Systems Conference in Fort Worth on November 14, 1989. A subsequent demonstration of the MDHC BHTI long haul connection to US Army personnel took place on February 9, 1990. This time the audience was in Mesa, Arizona. The two helicopters went on a reconnaissance mission together. The Mesa pilot discovered and painted an enemy helicopter. The Ft. Worth pilot then shot it down with a missile.

The MDHC gateway was subsequently upgraded to communicate at 56 Kilobaud using dial up Switch 56 service.

VI SUMMARY

The networking needs of the MDHC distributed real time system are met by a combination of two local networks. Real time communications is done by a high speed token ring Pronet 80. Non real time services are provided by an Ethernet with TCP/IP based protocols. Long-haul networking is achieved by interfacing a local network to a long-haul service in a Gateway Computer.

The original Ethernet Stealth Protocol allows transparent "peeking" and "poking" in memory of running targets. The non interference nature of the Stealth Protocol together with the separation of physical networks together segregates the data collection process. Data collection cannot impact the integrity of the real time simulation.

REFERENCES

1. ProNET-80, 80 Megabit/Second Local Area Network Doc #068610 by Proteon.
2. A. Katz, D. Allen, and J. Dickson, "Synchronization and Time Tagging in a Distributed Real Time Simulation", ALAA Simulation Technologies Conference, Boston, August 1989 p 259. To appear in Journal of Aircraft, August 90.
3. A. Katz, "Absolute Time Stamp in Networking of Simulators", position paper #008-01-90, Summary Report, The Second Conference on Standards for the Interoperability of Defense Simulations, Vol III, University of Central Florida Institute for Simulation and Training, January 1990.

NAVIGATIONAL AND ENVIRONMENTAL SIMULATION ISSUES FOR LARGE-SCALE NETWORKS

John Burnett, Section Head,
Software Engineering

Gary R. George, Senior Staff Engineer

Samuel Knight, Staff Scientist

CAE-Link Corporation, Link Flight Simulation Division
Binghamton, NY

Abstract

Networking of simulators may be the most revolutionary idea to hit the simulation industry this decade. Many companies today are involved in networking their own specialized simulators, which, in general, are designed to support the appropriate fidelity level of the devices they are interconnecting. However, the greatest training potential can be obtained if we pursue a long-term goal to have simulators of all fidelities, as well as embedded devices, interoperational through interconnectable long- and short-haul networks. This paper examines the navigation and environmental issues that present serious technical challenges in the development of combined arms and joint forces large-scale networks. The paper focuses on ways to provide uniform environmental models and simulation concepts for new simulators, and on the changes necessary to existing devices to support the large-scale networks foreseen in the very near future.

Introduction

The concept of the interoperability of all defense simulators in large-scale networks has recently been advocated. The "power strip" concept, where a simulator simply plugs into the net and becomes a player, is at best idealistic and simplistic. Issues of compatibility of fidelities of visual and sensor databases¹ have been considered. However, an often overlooked, but just as important, networking issue is the correlation of navigational and environmental simulations in networks of simulators.

Since many different types and fidelities of devices will be networked in the future, incompatibilities in environmental and navigational simulations are likely to exist between devices. Large-scale exercises will include air, ground and naval simulators with different environmental simulations. Incompatibilities may exist in varying degrees as a result of different modeling for earth, ocean, weather, magnetic variation, coordinate conver-

sions and wind, pressure, and temperature simulations.

Another subject of correlation is weather and cloud effects. Even in this area of simulation for single devices there has been little work.^{2,3} For large-scale exercises the effects of large storm fronts can have a marked effect on the overall mission. Hart⁴ in a recent paper pointed out that aircrews have used clouds and weather to their advantage (or disadvantage) for a long period of time. Flight simulators which are networked require weather simulations in a broad sense and correlated effects.

As team training objectives are developed and the use of networked simulators for advanced mission training and rehearsal becomes available, the concerns over correlation of environmental and navigational simulations between networked devices to support mission training and rehearsal will be particularly important. Not only must new devices specifically designed for networking be considered, but also the many existing devices in training today worldwide.

It has become apparent at working group meetings on networking that each group (land, sea, and air) has very unique networking requirements. This is especially true in regard to environmental and navigational simulations. For example, naval simulations are not concerned about land-based navigational facilities, just as the land-based close air support simulators do not need high-level ocean models. On the other hand, there are some fundamentals that are common, such as earth and global positioning modeling. This paper does not attempt to explore all the issues for all disciplines. It explores several issues from aviation simulation based on CAE-Link's experience with Multiple Simulator Networking (MULTISIM). Also, some options are presented for network implementation.

Radio Facility Data Base

Navigation sensors requiring external radio facilities include VOR/ILS, TACAN, and OMEGA, to mention a few. A common approach to simulating

these systems is to establish a radio facility data base containing facility pertinent data required for simulation.

This data must be easily accessible and usable in real time. This data also must be easily updated, including addition or deletion of stations as well as changing existing station data. Furthermore, this data may need to be changed quickly for mission rehearsal.

In order to accomplish this, many simulations use a special compiler to build or modify a file of radio navigation facility parameters in disk storage. This data is obtained from the DoD Flight Information Publications. The data in disk storage is usually different for various simulators. Facility type data is dependent on the aircraft navigation equipment and the mission of the simulated aircraft. This difference can obviously cause a problem with a number of networked simulators performing a coordinated mission. Navigational aids should be common for the various devices. The solutions to the problem are:

- (1) Change the navigational facilities file so that it is common for each device that requires navigational facilities. This would require a coordinated effort to review the data with a number of different contractors for each simulator. Also, facilities parameters are updated often and a continual coordinated update would be required.
- (2) Define unique navigational facility data in a similar manner to (1) but have it reside in a central point on the network with a unique node. Disadvantages in this approach include the requirement to change some indexing on each device for a common list and provide logic to use the central point data only in network mode.

Earth Model

In simulation the earth's surface can be represented by an ellipsoid of rotation around the earth's spin axis. To accurately model this, an ellipsoid is selected which best approximates the size and shape of the earth within the bounds of a particular gaming area. Gaming areas vary in size and location. A small gaming area at the equator could be approximated with a simplistic spherical model. Much larger gaming areas require a more rigorous ellipsoidal model. Differences in modeling on a network could produce inconsistencies in the present position computations of the vehicle, since position (latitude and longitude) is a function of an "initial condition" of present position which is integrated based on an object's inertial velocities ("navigation" frame reference) and the radii of

curvature of an ellipsoid. Ultimately these inconsistencies may cause degraded training for combined arms teams. This would include the inability of crews in multiple devices to accurately follow the same routes via waypoints, uncertainty of position relative to other simulators on the network, and for missions of long duration the ability of each device to arrive at the designated target or terminal area. Options for the earth model include:

- (1) The use of a central point on the network to provide all devices with a uniform model. Use of a unique network node or central point on the network would require each device to provide vehicle inertial velocities which could be integrated into the corresponding latitude and longitude as determined by the specific earth model at the central point. For the non-network mode, each device would revert to its own model. A major disadvantage of this approach is transport delays for positioning update data.
- (2) Change earth modeling to a uniform one for all simulators. A general trend on earth model simulation is the World Geodetic System (WGS)-84. This option is being considered by the Working Group on the Interoperability of Defense Simulations.⁵ Disadvantages of this approach include software changes at each simulator and definition of the most advantageous model.

Global Positioning System (GPS) Satellite Coverage

The use of the Global Positioning System for accurate navigation will be particularly important for deep strike penetration and special operations. The effectiveness of this system in the real world depends on the satellite coverage at the time of the mission. A common approach for modeling the GPS navigation solution is desirable. Options include instructor control of the navigation solution and modeling the navigation solution as a function of a satellite constellation.

For complex networked team missions (particularly unique mission rehearsal), the effects of satellite coverage and its relationship to navigation accuracy are necessary. The effect of one team being delayed (malfunctions, lost time, etc.) in a support mission and loss of accurate GPS requiring use of other systems must be considered.

Magnetic Variation

The difference between true and magnetic north varies at different earth locations. Differences in the way magnetic variation is defined between networked devices will cause navigational errors for intercept and coordinated attack.

Simulation of magnetic variation in the navigational environment is currently done in a number of ways:

- (1) A constant value of magnetic variation for simulations which use relatively small data bases.
- (2) Use of a spherical harmonic model (similar to the one used to produce the Naval Oceanographic Magnetic Variation Charts) to generate magnetic variation for a given location, and date.
- (3) Interpolation of data using a very large data base of magnetic variation values (in special files) for the entire planet or specific gaming area. The DoD publishes magnetic variation information for airport locations in Flight Information Publications.

The options available to get magnetic variation consistent in networked simulators are: 1) use of a general method or model at each device or 2) apply that same model at a central point on the network. A device with a unique node on the network would take each network device's position and determine the corresponding magnetic variation based on some model. Consideration of the team mission is necessary. Long-range mission rehearsal certainly cannot use fixed or constant magnetic variation.

Weather

Efforts to simulate full weather conditions in existing simulators have been limited and need to be expanded even for single devices. One method for large-scale simulation is pressure, wind, and temperature modeling, which has been applied by Link on several aircraft simulators.

Pressure, wind, and temperature simulation provides a weather environment in which pressure pattern navigation can be accomplished and in which pressure, wind, and temperature have a realistic effect on aircraft system performance. Correlated PWT simulation for networked simulators is necessary for team training. Naturally, the simulation varies among simulators based primarily on mission and simulated global position, including:

- (1) Constant values set at initial conditions or by instructor edits similar to existing simulators. Some may be altitude dependent, such as wind speed and direction at different altitudes.
- (2) Provide disk data on wind speed, wind direction, outside air temperature, and the true altitude associated with each of the pressure layers at a grid point. Interpolation is done at positions away from the specific points. This data is derived from

National Oceanic and Atmospheric Administration Worldwide data.

To supplement these large-scale effects, local conditions may be simulated, such as wind gusts, turbulence, wind shear, icing contrails, and microbursts from thundershowers as described by Klehr.² Weather radar simulations should correlate with the meteorological conditions which are simulated. Options to incorporate PWT into networking are similar to other navigational environment issues, namely to define a necessary model for the team training mission requirements and incorporate it at each device or at a central point on the network.

The local effects must also correlate and require uniform modeling in each device or at a central point on the network.

Other Environmental Issues

Simulations which include land and naval devices require correlations of environments that are significantly different from the aviation simulation community in some respects. The ocean models required by naval simulations must be correlated between networked devices in regard to both appearance and physical attributes. Appearance characteristics include sea state, ice flows, and mappings of the ocean bottom. Physical attributes include temperature and velocity profiles, and radar characteristics. The ground simulations require correlated visual data bases as well as physical information including terrain state and friction coefficients. Terrain state includes the correlated effects of dynamic terrain where the terrain can be altered dynamically by a tank or bulldozer as it operates in the data base.

The Options

The concept of a central point navigational and environmental simulation and modifications to existing devices have been presented as options for the problem of navigational and environmental simulations for large-scale networks.

The concept of a "Universal Navigation/Environmental Simulator" to provide a total simulation of physical attributes of the environment is a possible solution. This would become a unique node on the network and supply data to the various simulators in regard to their required environmental simulation requirements much like the Universal Threat Simulator that is also being proposed. Data not needed by a specific device (such as navigational facilities by certain naval and ground devices) would be ignored. Some obvious problems with this approach are use of bandwidth with this increased network traffic, time delays, and the adequate definition of the models. It may still be necessary to retain stand-alone models for single training or for cases when the central point is not in

operation. This concept may not be cost-effective in the near term but will be necessary for the large-scale training exercises using networks planned for the next century.

The alternative for today is to have necessary correlated models in separate devices on the network. This would require coordination among several contractors in doing updates to existing simulators and providing specifications on environmental/navigational modeling visual database correlation. Little research has been done to define fidelity differentials and resolution between devices in regard to the network data. This approach can be implemented today by defining the necessary protocols and model requirements. This process has just started with the first networking standard draft.

Conclusion

Correlated data bases of all types will be necessary for large-scale networking. This paper has examined several issues of environmental and navigational data base correlation, primarily in regard to the aviation simulator community, although the options to provide successful large-scale networking are similar.

There is a need to define the necessary environmental/navigational requirements for each of the land, sea, and air simulation communities. The would be part of the mission-critical task analysis for each group in regard to networking. A difficult part of this analysis will be the required interaction

between devices in regard to the navigational/environmental issues mentioned in this paper.

In order to train war fighters using large-scale networks, it is essential that they can navigate to their necessary mission position and experience common environmental factors that are correlated to allow for realistic simulation of their simulated mission.

References

1. Hrabar, M., Joosten, John, Widder, P.A. "Data Base Conversion/Correlation Issues" I/ITSC 1989.
2. Klehr, J.T. "Simulation of Hazardous Flight Conditions" AIAA 89-3303-CP; Flight Simulation Conference 1989.
3. Handberg, G.O. "Meteorological Inputs to Advance Simulators" Proceedings 5th Annual Workshop on Meteorology; December 1981.
4. Hart, B.H. "Fleet Requirements for F-14D Aircrew Trainer Suite" AIAA 89-3303-CP; Flight Simulation Conference, 1989.
5. "Summary Report — The Second Conference on Standards for the Interoperability of Defense Simulators" The Institute for Simulation and Training, January 15, 1990, Orlando, FL.

INTERFACING LOW COST NETWORKED FLIGHT SIMULATORS IN A SIMNET ENVIRONMENT

Jorge Cadiz
Ruey Ouyang

Margaret Loper
Jack Thompson

Institute for Simulation and Training
University of Central Florida
Orlando, Florida 32826

ABSTRACT

In this paper we present a discussion of research being carried out at the Institute for Simulation and Training/University of Central Florida which focuses on the interconnection of dissimilar networkable simulators. Specifically, we describe the results of our efforts to interconnect two dissimilar simulation networks, namely, the Perceptronics Avionics Situational Awareness Trainer (ASAT) networkable F-16 training device and the DARPA developed SIMNET simulation network.

INTRODUCTION

The advent of direct computer to computer communications (computer networking) opened the possibility of interconnecting many different types of computer based systems. Until recently, most devices used in Simulation and Training (S&T), which usually contained an embedded computational resource, operated in a stand alone mode. Today there is a major emphasis being placed on the development of distributed S&T systems which are **networkable**. In this context, *networkable* implies the S&T systems are capable of communicating (transmitting and receiving), in real-time, information relative to their simulation which can be understood by other networkable devices tied to the network, thus allowing for interaction between the devices. However, due to the lack of a standard network protocol, the interconnection of multi-vendor simulators on the same network presents some interesting problems.

Institute for Simulation and Training (IST) is presently under contract to DARPA and the Army's PM TRADE to develop a low cost Aviation Test Bed to investigate issues in low cost aviation simulators for training and the networking of these types of devices. IST has procured two commercially available F-16 ASAT devices from Perceptronics. The ASAT is a PC based (MS-DOS) training device developed to teach beyond visual range (BVR) target acquisition and engagement. The simulator can be operated in a stand-alone mode or networked with other ASATs to provide a multi-ship, interactive training environment. In addition, IST has two M1 Tank SIMNET modules, along with other SIMNET support modules. Currently in the research and development stage, SIMNET is a networked simulation environment made up of armor, mechanized infantry and aircraft (rotary and fixed wing) simulators linked together such that actions of one vehicle can be observed by the

crews of the other vehicles. Both the ASATs and the SIMNETs utilize ETHERNET local area networks to communicate vehicle state information between the simulators engaged in the training exercise.

In fulfillment of the requirements of the research being performed by IST, the task of interconnecting the ASAT network with the SIMNET network was undertaken. This task centers around the design and development of a device referred to as a Network Protocol Translator (NPT). The NPT device will act as an interpreter between the ASAT and SIMNET networks; receiving data from the ASAT network, transforming it into the appropriate SIMNET format, and then retransmitting it onto the SIMNET network, and vice versa. In order to do this task correctly under real-time constraints, a detailed knowledge of the exact workings of each simulator's network is required.

Although both the ASAT and the SIMNETs are networkable and use ETHERNET as a communications medium, the information (packets) transmitted by one simulator is totally unrecognizable by the other in its native form. Simply stated, the task of providing interoperability between the two simulators involved the translation

of information contained in the ASAT ETHERNET packet into the appropriate SIMNET ETHERNET packet format and vice versa.

APPROACH

To achieve networked communications between the two simulators, a protocol translator was placed as an additional node on the ETHERNET network (see Figure 1). Functionally, the translator monitors the network traffic and copies packets which are transmitted by the ASAT trainer into its local memory. Every packet copied by the protocol translator is converted to the proper SIMNET format and transmitted onto the ETHERNET. The SIMNETs can now receive these packets as they would any other standard SIMNET packets, and act on it appropriately (i.e., display aircraft on local simulator's visuals).

A similar approach for SIMNET to ASAT protocol translation can be envisioned. However, due to the nature of the ASAT hardware/software configuration, packet translation in this direction would not have been possible without some modifications to the ASATs themselves. Since one of our goals was to provide a passive protocol translation system this translation was not fully implemented.

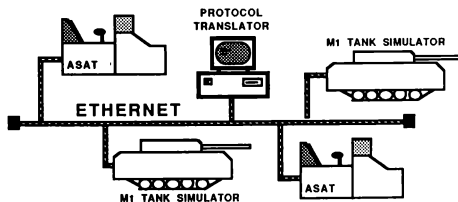


Figure 1. ASAT/SIMNET Network

PROTOCOL TRANSLATOR

The protocol translator was developed on a 20 Mhz 80386 IBM/AT compatible PC. To connect it to the ETHERNET network, the PC was outfitted with a 3Com Etherlink II Network Adapter. This adapter was configured to operate in promiscuous mode such that it would be able to capture all packets transmitted onto the network. The translator would filter all unwanted packets, taking only the properly addressed packets into its memory for processing.

ASAT PROTOCOL

There are four different types of packets that the ASAT uses to initialize and execute an exercise:

- Packet Type 1: is used during initialization of an exercise.
- Packet Type 2: activates the vehicles with their proper parameters (e.g., position, orientation, speed, weapons' capabilities)
- Packet Type 3: provides information regarding the aircraft's location, elevation, orientation, airspeed, and other state variables.
- Packet Type 4: provides information regarding the aircraft's location, elevation, orientation, airspeed, and other state variables for all computerized targets.

The information contained in Packet Type 3 (i.e., location, elevation, orientation, airspeed) were the main parameters used in the ASAT to SIMNET protocol translation algorithms.

SIMNET PROTOCOL

The communications protocol used by the SIMNET network is the SIMNET Protocol Version 6.0 which is described in [1]. The SIMNET protocol has three subprotocols: Data Collection Protocol, Association Protocol, and the Simulation Protocol. The only protocol of interest in the translator experiment was the Simulation Protocol.

The Simulation Protocol provides the necessary tools for allowing a simulator to describe any of its actions that may affect other nodes participating in the same exercise. Several of the functions that are furnished by the simulation protocol are:

- Activation and Deactivation
- Vehicle Appearance Updates
- Vehicle Status
- Weapons' Effects and Interactions
- Detection and Reactions to Vehicle Collisions
- Service and Repairs to Vehicles

The Vehicle Appearance Updates are provided by the SIMNET Vehicle Appearance Protocol Data Units (VA PDUs) and were used to provide the ASAT information to the SIMNETs.

TRANSLATOR PROCESSING

The translator creates a template of the SIMNET VA PDU for each ASAT packet that is to be translated. This template is supplemented by translated ASAT data to create a complete VA PDU for the ASAT. The fields which are supplied by the ASAT packet are:

- Source Address
- Vehicle Location
- Vehicle Speed
- Orientation (roll, pitch, and yaw angles)

Mapping the ASAT supplied information into the SIMNET VA PDU format required the following:

Source Address - ASAT 6 byte ETHERNET source address mapped directly into the SIMNET VA PDU source address field.

Vehicle Location - Coordinate transformation required to provide proper unit matching and a compatible location in the SIMNET terrain database.

Vehicle Speed - Manipulation required to convert ASAT airspeed data into SIMNET normalized velocity components (x, y, z velocity). Unit conversions were also required.

Orientation - Manipulation required to transform ASAT pitch, roll, and yaw angles into a nine element SIMNET rotation matrix.

(Note: A detailed description of the translation algorithms can be found in reference [2].)

SIMNET TO ASAT TRANSLATION PROBLEMS

In trying to network the ASAT and SIMNET simulators, several complications prevented their complete interfacing. The first problem resulted from the inability of the SIMNET vehicles to be properly initialized in the ASAT environment. Upon activation, the ASAT trainers undergo a handshaking process. From this

process each ASAT creates a list of all participants in the exercise. If a vehicle is not involved in this initialization procedure, it will not be a participant, and ASAT simulators which are a part of the exercise will ignore any packets received with the source address of that vehicle.

The inability to perform the required handshaking and thus be placed on the ASAT players list made it impossible for the SIMNETs to be initialized into the ASAT exercise via the Protocol Translator. In the SIMNET environment, it is not critical for this function to be performed. If a vehicle is not activated properly, the other vehicles on the network will proceed to acknowledge the vehicle's existence in the exercise and allow it to interact with other players.

Other complications arose because the ASATs were designed to train pilots in air-to-air BVR techniques. One of the most prominent problems was that the ASATs do not support vehicles which have zero velocity. Because the SIMNET MIs are ground vehicles, they operate at low velocities and are static in many situations (i.e., zero velocity). These scenarios could not be duplicated in the ASAT environment without modifying the ASAT source code. The design requirements of the ASATs did not call for the existence of ground vehicles. Therefore, no models for vehicles such as tanks are provided in the ASAT environment. The only moving models that are provided by the ASAT visual system are an F-16, a Mig21, a Mig29, and a generic missile.

Another problem was the ASAT addressing scheme: its protocol is not of a standard IEEE 802.3 format.

The ASATs begin their packet with the source address immediately followed by data. This deviation from 802.3 format caused difficulty in data analysis using standard ETHERNET LAN Analyzers.

TRANSLATOR PERFORMANCE TESTS

There are several considerations when translating a simulator's packets through a protocol translator. Two factors which are of importance are the amount of increase in traffic on the network, and the transmission delay due to processing the ASAT packets.

The increase in network traffic must be considered a factor in networks which have a limited bandwidth/simulator ratio. The increase in traffic is proportional to the number of simulators which are dissimilar to the standard network. In our experiment, we had one simulator anomaly (ASAT) which was connecting into the SIMNET network. The increase in traffic was given by the simple formula:

$$\text{Increase in traffic [one-way translation]} = N_{pt}(R_A)$$

$$\text{Increase in traffic [two-way translation]} = N_{pt}(R_S + R_A)$$

where N_{pt} is the number of protocol translators on the network, and R_A and R_S are the rates of transmission for the ASATs and the SIMNETs, respectively.

In our experiment we had a single protocol translator performing a one-way translation. We are assuming that one translator will translate the traffic of a single ASAT simulator. The increase in traffic is equal to adding another ASAT module onto the network.

This means that on the average the increase will be:

$$(1 \text{ translator})(113 \text{ bytes/pkt})(12 \text{ pkts/sec}) = 10.848 \text{ Kbits/sec}$$

This increase in traffic is merely 0.15% of the usable network bandwidth, based on 7 Mbits/sec as the usable ETHERNET bandwidth. However, it must be kept in mind that the ASATs are selective fidelity simulators and do not have as rapid an update rate as most flight simulators.

Transmission delays are a critical factor in the implementation of a protocol translator. In a distributed simulation environment, the systems can ill afford an extra delay produced by a protocol translator. In this light, we have conducted timing tests on the translator. These tests have produced statistics about the total translation delay introduced by the ASAT/SIMNET translator. The translation of the ASAT packet to a format which the SIMNETs can recognize caused an average network delay of 29 ms. This included time of packet processing at the board level.

CONCLUSIONS

Herein we have discussed the various aspects of IST's research involving the networking of dissimilar distributed simulation and training devices: specifically, interconnecting the ASATs and the SIMNETs. Impacts on network loading and increases in network delays were discussed and experimental results were presented. Insight gained from this research will provide the basis for the development of a Generic Protocol Translator device which would be able to translate data packets between various types of dissimilar simulators. However, the

problems stemming from the interconnection of high and low fidelity simulators, aside from basic communications issues, are beyond the scope of this work and need to be addressed in full detail.

ACKNOWLEDGEMENTS

This work is supported by the U.S. Army Program Manager Training Devices (PM TRADE) and the Defense Advanced Research Projects Agency (DARPA) under Broad Agency Announcement #88-01, contract number N61339-89-C0043. The views and conclusions herein are those of the authors and do not represent the official positions of the funding agencies, the Institute for Simulation and Training or the University of Central Florida.

The authors would like to thank Graduate Student Assistants Mike Ruckstuhl and Gilbert Gonzalez for their help in software development, obtaining and analyzing network data.

REFERENCES

- [1] Pope, A., "The SIMNET Network and Protocols." BBN Report Number 7102, (July 1989).
- [2] Cadiz, J., et. al., "ASAT to SIMNET Protocol Translator Hardware and Software Description." IST Technical Report Number IST-TR-90-10, (June, 1990).

NETWORKED MODULAR AIRCREW SIMULATION SYSTEMS

David E. Powell, Dr. James W. Dille & Steven D. Swaine
McDonnell Aircraft Company
A Division of
McDonnell Douglas Corporation

Stephen M. McGarry
Advanced Simulation Division
Bolt, Beranek and Newman, Inc.

Abstract

Large multi-vehicle simulation and training environments are increasingly necessary as the focus of aircraft simulation shifts from the individual to the team, both for the development and testing of advanced aircraft systems and for effective aircrew training. Budget constraints dictate that a solution must incorporate both existing simulators and new low cost reconfigurable simulators that can serve as a variety of aircraft types. A developing network standard, based on the SIMNET protocols, will allow for the interconnection of the number and variety of simulators required. A recent experiment, the use of SIMNET to network high performance aircraft simulators at the 1989 I/ITSC, has provided valuable insights into the problems faced when interconnecting existing simulators via SIMNET.

Introduction

The intent of this paper is to discuss simulation networking issues, particularly as they pertain to the integration of existing simulators into a SIMNET based network environment. Towards that end, a certain amount of time will necessarily be spent discussing the simulator systems themselves, their architectures and capabilities. In this networking effort, the Modular Aircrew Simulation Systems (MASS) developed by McDonnell Douglas figured prominently. After a brief discussion of the MASS systems, the remainder of the paper describes the cooperative networking effort undertaken by BBN and McDonnell Aircraft Flight Simulation, an air combat demonstration shown at the Interservice/Industry Training Systems Conference (I/ITSC) held in November 1989 at Fort Worth, Texas. For this demonstration, two manned MASS simulators, an F-15 and an F-18, and a digital threat environment

were networked via BBN's SIMNET protocol and interfaced with other SIMNET equipment on the show floor.

MASS Overview

With large scale multiship manned simulations becoming increasingly necessary, it was clear that the technologies currently in use were inadequate to meet this need in a cost effective manner. New techniques were needed to lower the cost of simulation while preserving fidelity and performance. Four major technology areas were identified; simulation networking, computer architectures, reconfigurable cockpits, and low cost visual systems. Progress in each area has lead to successive generations of MASS Systems. Two previous papers presented at the 1988 and 1989 AIAA Flight Simulation Technologies Conferences have described these systems in some detail. Summarized briefly below are the four technology areas and the current state of progress in each area.

Computer Architectures

In order to take advantage of the latest commercially available hardware and software products, the MASS systems adhere as much as possible to an open systems approach. Figure 1 shows a block diagram of the F-15 simulator. MASS computer systems are presently VME based, with primarily 68030 based Motorola MVME147 boards and 88000 based MVME188 boards as target processors. Code development and loading of the targets is accomplished from a resident UNIX system running on one of the MVME147 boards. Other boards in the system include digital and analog I/O boards, sound generators and graphics image generators.

Unix does not take part in the real time execution, nor is there a kernel operating system on the targets for both performance and cost reasons. System software is written in both C and assembly language, while most of the application software is in Fortran. Both synchronization of the parallel microcomputers and data transfer between them is through a message passing protocol implemented on the VME bus.

Reconfigurable Crewstations

Often using flight hardware and dedicated to a single use, the physical crewstation has traditionally been a high cost item in an aircraft simulation. MASS systems supply fidelity only where required, for example using flight hardware sticks and throttles, and populating the side panels for only the functions required by the mission. The main instrument panel is computer generated and presented to the pilot via a 27" direct view high resolution monitor equipped with a touchscreen for knob and switch actuation.

Figure 2 shows a photograph of the F-15 MASS system. With a reload of the host computer, the replacement of stick and throttle grips and the removable side consoles, it can change rapidly from one type of aircraft to another. For the simulation of various multitship scenarios, this offers a valuable economy. It also allow such cockpits to be shared by various aircraft programs, for the cockpit can serve a variety of purposes and remain unclassified after a particular use.

Low Cost Visual Systems

MASS system visual requirements include the main instrument panel display, HUD, and the out the window scene. Depending on the aircraft being simulated, the main instrument panel and HUD are generated by either Io Inc. board sets, or by diskless Personal Iris board sets. In either case, the image generation equipment is contained within the crewstation, along with the computational host.

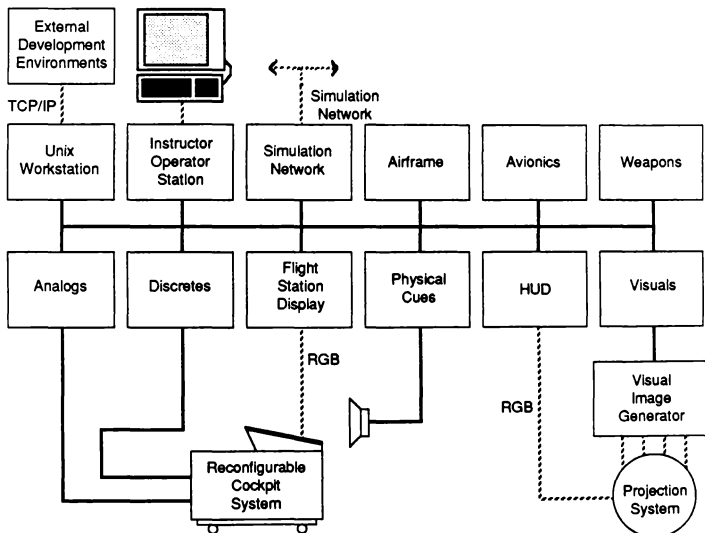


Figure 1: MASS F-15 Block Diagram

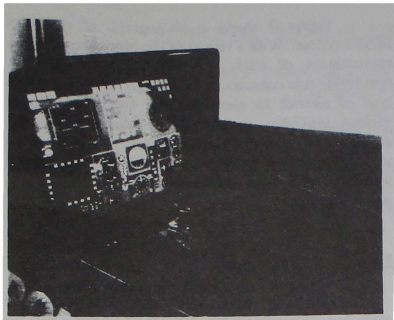


Figure 2: MASS F-15 Crewstation

External to the cockpit is the out the window visual image generator, typically multiple channels of Paragon. Figure 3 shows the four flat panel display system currently used by the MASS systems. This low cost projection system provides approximately a 200 degree horizontal by 105 degree vertical field of view for the pilot. Electrohome ECP-3000 projectors are used and a monochrome projector mixes the HUD onto the center channel. The HUD is generated and projected separately instead of imbedded in the center channel to achieve sufficient HUD resolution.

Simulation Networking

As will be explained in much greater detail below, the decision was made to evaluate the SIMNET networking approach to see if it could perform to the requirements of high speed combat aircraft with advanced avionics. At this time, SIMNET is the only networking approach used by the MASS systems.

1989 I/ITSC Conference

Background

In mid 1989 MCAIR was constructing three MASS systems, one configured as an F-15, one as an F-18, and one as a digital threat environment. The F-15 was contracted to be delivered to the Human Resources Laboratory at Williams Air Force Base (AFHRL), and the F-18 was to be a technology

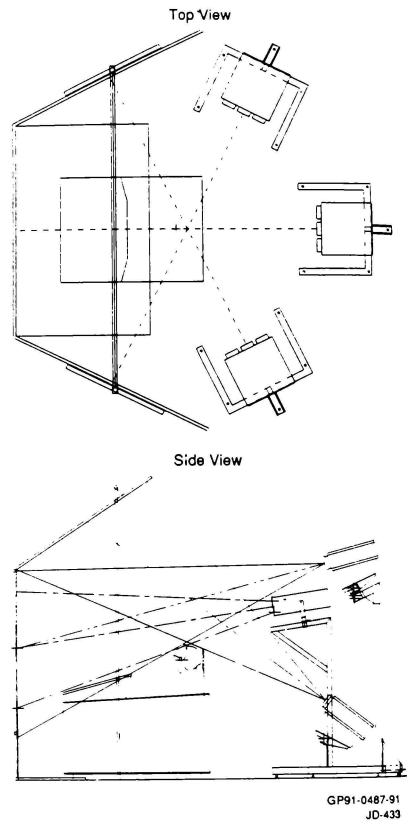


Figure 3: Four Channel OTW Visual System

demonstrator for the McDonnell Training Systems Company. Having taken a MASS system to several previous I/ITSC shows, it was decided to take all three systems to the 1989 I/ITSC, network them together on the show floor, and fly 2 v 8 air combat scenarios.

MCAIR had previously networked at the 1988 I/ITSC, with an ethernet link using a MCAIR developed protocol. This network was used to connect a MASS cockpit in the McDonnell Douglas Booth to another MASS computer system which was driving a crewstation belonging to the Paragon Graphics Company and located in their booth. At the same I/ITSC, BBN was demonstrating their SIMNET networking with several tank simulators and long distance links.

With the success of the original SIMNET program, it was apparent that the protocol would be extended far beyond the armor and close air support simulations it had previously served. It remains clear that the SIMNET protocol will become the core of a government/industry standard for the interconnection of simulators. MCAIR was interested in evaluating the suitability of the SIMNET approach for the networking of high speed aircraft with full avionics and electronic warfare capabilities. It was not apparent from the previous applications of the protocol that the performance for air to air combat simulation would be acceptable.

At the same time, BBN had entered into a contract to network existing trainers at AFHRL, enhancing the SIMNET protocol to handle high performance aircraft, which would result in the Air Forces ACMENET standard. With the F-15 MASS system deliverable to AFHRL, it would be one of the systems that BBN would have to network together. It would benefit BBN to get as early an exposure as possible to the problems that would arise when high performance aircraft were brought into a SIMNET network environment. BBN was also interested in the lessons that could be learned from what would be the first networking of a device that was not designed and built for SIMNET by one of the original SIMNET contractors.

It became clear during the planning for the 1989 I/ITSC, that a cooperative networking demonstration would be beneficial for both companies. With Paragon Graphics joining in to supply the out the window visual image generators for both of the MASS crewstations, an agreement was reached between the three companies that would lead to a rapid prototype of a SIMNET compatible air combat network. BBN would network the two MASS crewstations and the digital threat generator, leading to the demonstration of 2 v 6 air to air combat scenarios.

System Architecture

As BBN integrated SIMNET into the MASS systems in St. Louis, the overall system consisted of the F-15 and F-18 cockpits and the threat generator. As indicated above, MCAIR's MASS systems are VME based with UNIX as the development environment and running on bare target boards. BBN utilized SUN workstations for development and while their code also executed on Motorola MVME147 boards, they utilized a kernel operating system. It was decided for this demonstration, that instead of BBN furnishing entire interface units, that the SIMNET code would be adapted to run within the MASS system architectures. As indicated in Figure 1, a MVME147 card was dedicated to the SIMNET interface.

The SIMNET code underwent some alterations to remove its dependence on its kernel operating system such as memory allocation calls. MCAIR supplied ethernet routines altered to adhere to the SIMNET standard and code was written to integrate the MASS message passing protocols to the BBN code.

A previously defined communications array which contains the current state of the aircraft was message passed from the Avionics board to the SIMNET board once every 20 Hz frame. The SIMNET board took what information it required from this array and according to its protocol, broadcast it onto the network. Similarly, the board would receive ethernet packets and update the states of the other aircraft, supplying that information to its host upon request, again once per frame.

The SIMNET interface also made provisions for flying out the missiles as separate bodies, obtaining the missile flight information from the aircraft who fired the missile. The threat generator provided a slightly more complicated problem, as one SIMNET board served as the interface for the six digital threats.

Network Integration and Checkout

With the architecture as defined and the SIMNET code rehosted onto the MASS systems, two tasks remained. They were to make the network function between the three systems and to verify that the network traffic was indeed SIMNET compatible. For the latter task, BBN furnished a MassComp which functioned as a data logger. Tapes of ethernet traffic could be created during tests of the MASS network and taken to Cambridge where they could be replayed and

between the exercises. The highlight of the three day demonstration came when the exercises were merged and the long haul link established to the Semi-Automated Forces in Cambridge. Observers in Cambridge could watch the air battle and direct their forces. Air defense vehicles armed with surface to air missiles were thus brought into the air combat scenario and succeeded in shooting down several of the MASS aircraft, both manned and unmanned.

There were several difficulties from the MASS side during this unplanned demonstration. The Paragons had no proper air defense vehicle models to display, so the demonstration was most effective from the BBN booth where the planes could be seen flying by while the air defense vehicles tracked and fired. Most discouraging for the MASS pilots was that the F-15 had no air to ground capability and while the F-18 did, this demonstration was not foreseen and the SIMNET interface code was not capable of sending out kill information for anything but the MASS vehicles. This shortfall was corrected overnight and the F-18 did destroy several air defense vehicles the next day when the long haul link was re-established.

Lessons Learned

The network demonstration was widely seen to be a success, with pilot acceptance immediate and largely uncritical. Issues attributed to the SIMNET approach and discussed below were noticed only by those looking for them. Various observations and the lessons learned are summarized below.

SIMNET Interface Utilization For the rapid prototype exercise, the majority of the overhead associated with converting to the SIMNET protocol at the MASS system interfaces consisted of

- Conversion of units
- Coordinate system conversion
- Status flag conversion
- Dead reckoning of the vehicles
- Message passing to/from SIMNET board
- Ethernet packet handling

The observed loading of the 25 Mhz 68030 SIMNET interface board was less than 2.25 milliseconds per vehicle at any time during the demonstration. Since the processing of the received information dominates, we can extrapolate that the MASS systems using the current hardware could have handled a total scenario size of approximately 20 aircraft.

Bandwidth Considerations It became clear that with any acceptable error criteria, the MASS aircraft would often be sending packets at the maximum rate of 20 Hz. For example, a sustained turn would hold that rate of transmission. In fact, the manned systems were typically sending every frame except when they attempted to formation fly. The digital threats were smoother flying except when under attack and that contributed to a network average of 8 packets per vehicle per second. It was clear that in the midst of an engagement that all aircraft were sending at the maximum rate. A much larger scenario would be required before statistical arguments could be used to justify a reduction in bandwidth requirements due to the SIMNET dead reckoning approach.

Error Thresholds The original SIMNET dead reckoning error thresholds turned out to be inadequate when the pilots flew in formation. The rotational error criteria was originally set at 3 degrees and the translational at 10 percent of the body dimension along each axis. When flying formation, the other plane as viewed in the side channels of the visual system would occasionally jump forward or backward by this 10 percent amount. When flying smoothly as required to stay in formation, the algorithms were able to save on network bandwidth at the expense of the visual fidelity, ironically doing so when that loss in visual fidelity was most noticeable.

Although operating correctly according to the protocol, this jitter was determined to be excessively distracting. On the show floor, the rotational error criteria was reduced to 1 degree and the translational error criteria was reduced to 5 percent, reducing the problem although it still remained noticeable. The problem had not been foreseen during the checkout in St. Louis, where only front channels of out the window image generation were available. This phenomena has inspired a proposal to make the error criteria dynamic, so that when attempting aerial refueling or formation flying, a vehicle can request another vehicle to temporarily increase its rate of position updates.

Differing Frame Rates The MASS vehicles ran at a 20 Hz. frame rate while the BBN devices operated at 15 Hz. This skew in the frame rates manifested itself in an anomaly observed in the BBN observation vehicle and noticed by a number of people. This was a beat caused by the two frame rates as two position updates would occasionally be received within a single frame on the observation vehicle. An algorithm for smoothing this jitter was employed on the BBN device which eliminated the problem.

Database Correlation The MASS simulators used a different database from the SIMNET devices which resulted in several accommodations. None of these problems were considered serious and most could have been alleviated through database correlation had time allowed. In order for the MASS devices to appear on the Plan View Display, an offset vector needed to be added and removed from the position data by each MASS system. The combat scenario being used by the MASS systems was centered about an airport, with a point on the runway being 0,0 in a positive and negative coordinate system. SIMNET uses an all positive coordinate system and the SIMNET coordinates of an airport within the BBN database were used as the offset values. With the air combat scenarios used, pilots needed to pay little attention to any ground details except the airport, and this rudimentary correlation of airport positions allowed the observers at the BBN booth to follow the action quite well, without any correlation of the terrain data base.

More minor problems were encountered with the BBN Observation Vehicle, which operates by bringing in surface terrain as it comes within a certain range of the current observation point. When this point is attached to high flying aircraft such as the MASS vehicles, no horizon was visible. In order to obtain a horizon reference for better audience viewing, the typical altitude of the air combat engagement was lowered by commanding the digital threats down to 3,000 feet, a low altitude for air combat. A final minor problem was that the BBN observation vehicle had limited aircraft models and could display the MASS vehicles only as A-10s.

Jeff I. Cleveland II, Steven J. Sudik, and Daniel J. Crawford*
NASA Langley Research Center
Hampton, Virginia 23665-5225

The increased complexity and higher performance of modern fighter aircraft impose a requirement for more powerful tightly-coupled computer processors to calculate the requisite real-time simulation models. To minimize system time delays which introduce errors and false eigenvalues, it is desirable that the power be provided by a single processor or, if that is impractical, by multi-processors operating from a common primary memory. The customary approach in the industry is to employ multiple computers to simulate a system of associated aircraft. This in turn requires a wide bandwidth data communications system with low latency times to connect the various hardware elements of the simulation, including the computers. At the present time, most of the computers in the flight simulation industry appear to be near obsolescence. The NASA Langley Research Center, in upscaling the computer power by more than a factor of eight, has been dealing with these and other problems. In December 1989, the Center awarded a contract for an upgrade to simulation processors and the staff is presently integrating this upgrade into the facility. This paper reviews the requirements and proposed response, planned implementation, probable areas of difficulty, and current status and plans.

INTRODUCTION

An unpublished survey of flight simulation users at the NASA Langley Research Center (LaRC) conducted in 1987 projected that computing power requirements would increase by a factor of eight over the following five-year period. Although general growth was indicated, the pacing discipline was the design testing of high performance fighter aircraft. Factors influencing growth included: 1) active control of increased flexibility, 2) less static stability requiring more complex automatic attitude control and augmentation, 3) more complex avionics in general, 4) more sophisticated weapons systems, and 5) the need to simulate multiple aircraft interaction, the so called "n on m" problems. This requirement for more computing power is, if not industry wide, at least common to the fighter aircraft segment.

At present, there are no standard precise means for measuring net computing power. In practice, an LaRC benchmark program (CPU processing only, no input/output) based on an X-29 aircraft simulation is used to specify and test power. To associate computing power to more familiar but less precise units, the Digital Equipment Corporation (DEC) VAX 11/780 is used as a base and is assigned a power of one unit. In these units, a VAX 8650 scored about 6 with the benchmark and the Control Data Corporation (CDC) CYBER 175 (circa 1976) scored about 10. The Gould Concept 32/9780, the workhorse of the industry, also scored about 10. Using these ratios, the projected (and specified) computing power

requirement at LaRC was 160 times that of a single VAX 11/780.

The performance described above (with the exception of the CYBER 175, a 60-bit machine) was measured using a 32-bit word length. However, an important subset of LaRC simulations, including the X-29 benchmark, require greater precision than afforded by a 32-bit word. Therefore, performance was specified at 14 decimal digits for a floating point mantissa. This combination of precision and power significantly limited the field of potential vendors and in fact caused a one year delay in the acquisition process while the simulation computer state-of-the-art advanced. The Gould NP2 line and the ELXSI 6460 were dropped from development during this period. In addition, during this turbulent period the advent of Reduced Instruction Set Computers (RISC) attracted industry attention with the promise of low cost commodity computer power which included in at least one case (Intel 80860), 64-bit word length. At the time of vendor selection, RISC architecture had not been significantly integrated into the real-time simulation industry.

Two other factors which entered the considerations were vector processing and parallel processing. Vector processing, which lends itself nicely to the solution of the partial differential equations of computational fluid dynamics and structural design, has not yet been applied in a significant way to the ordinary differential equations of real-time simulation. Several areas in the real-time frame are obvious candidates for vectorization, such as numerical

* Senior Member AIAA

integration, external data communications, and especially function evaluation (look-up and interpolation) which uses about 40% of the required compute time per frame. However, simulation is primarily a scalar process and none of the vendors used vector processing to enhance their X-29 benchmark performance.

The potential of parallel processing (especially massively-parallel) is impressive. The attraction is low cost processors, such as RISC, which are combined in arrays to deliver extraordinary net power to the user who is seemingly unaware of the complex architecture. Hardware arrays have been built and computer engineers and analysts are now developing operating systems and languages to manage these arrays. Meanwhile, operational parallelism is restricted to a small number of processors per machine and to multiple functional units within a processor. Although some vendors offer automatic management of multiple processors, the workload imposed on the application analyst in the dynamic operational environment at LaRC must be minimized. To control the multiprocessor management burden, a single-processor power specification was included.

Another major factor to consider in all real-time systems is overall responsiveness. In combination with compute power, system response constrains minimum frame time and therefore controls the scope of systems which can be simulated in real time. The numerical integration algorithms (normally a second-order Adams-Bashforth predictor) used at LaRC and other facilities require at least 20 steps per cycle for acceptable accuracy. Normal frame times at LaRC range from 20 to 50 milliseconds. However, some modern fighters require 12.5 milliseconds frame time. Recently a hard requirement has surfaced for a one millisecond frame time associated with control of oscillations in a large space structure. This latter simulation application is an example of a prototype hardware-in-the-loop real-time simulation as opposed to the more normal (at LaRC) piloted simulation.

The requirement for responsiveness is best envisioned by considering this worst case. In one millisecond, sensor data must be converted and read into the computer, the control response must be computed, and then the computed data must be transmitted to signal converters and on to the actuators. This type of response requires wide bandwidth deterministic data communications with near zero latency in both the hardware and software.

Two characteristics of the LaRC simulation facility played an important role in formulating requirements. The first characteristic is the operational environment. At LaRC several independent simulations run concurrently. The simulation studies being run on one day may be entirely different from the simulation studies being run on the next day. Source code is subject to frequent change. Second, unlike the majority of U.S.

facilities, LaRC employs centralized computers as opposed to distributed computers. However, the assertion is somewhat misleading since in addition to the two large central scientific computers, many special purpose mini- and micro-computers are used at the simulation sites. Examples include the Evans and Sutherland CT-6 computer generated image system and the Symbolics artificial intelligence computer. In 1987, a data distribution and signal conversion system, referred to as the Advanced Real-Time Simulation System (ARTSS), was put into service at this Laboratory. The ARTSS has been very successful and is described in references 1, 2, and 3. Many of the response requirements, hardware and software, described in this paper concern the interaction with ARTSS.

Two single processor CYBER 175 computers, tightly coupled through extended memory are used to support flight simulation at LaRC. Having decided to continue using centralized computers, LaRC issued a Request for Proposals in May, 1989 and subsequently awarded a contract to CONVEX Computer Corporation in December of that year. The result of this activity is referred to as the Flight Simulation Computing System (FSCS). This paper reviews the requirements and proposed response, experience to date, planned implementation, probable areas of difficulty, and current status and plans.

REQUIREMENTS

In 1987 LaRC conducted a survey of simulation users and program managers to determine future requirements for flight simulation at LaRC. Results from this study (see Figure 1) indicated that high performance aircraft with expected increasingly complex models and flexible airframes would require up to eight times the model computing capacity compared to the present two CYBER 175 computers. These results coupled with other information, led to the definition of requirements for replacing the existing computers. Following an extensive survey of the marketplace, further action was delayed for one year to wait for development of systems that could meet the requirements. In 1989 the requirements were incorporated into a Statement of Work and a formal Request for Proposals was issued.

CPU Performance

Real-time flight simulation at LaRC requires high scalar CPU performance to solve the equations of motion of the system being simulated. Using an existing X-29 simulation as a benchmark, the following CPU performance is specified:

1. If a single CPU configuration is provided, the CPU must solve the benchmark in at most 165 seconds.

2. If a multiple CPU configuration is provided, each CPU must solve the benchmark in at most 330 seconds.

The capabilities of the resultant total system will provide eight times the CPU processing power of the coupled CYBER 175 computers.

Real-Time Input/Output

The ARTSS CAMAC (Computer Automated Measurement and Control, see references 4 and 5) system has provided LaRC with a high performance real-time input/output system that has extended the capabilities of the LaRC simulation system. Having a high transfer rate with low latency input/output system to support research simulation, LaRC desired to retain the ARTSS CAMAC system. LaRC requires that the new system include all software and hardware to connect to the ARTSS CAMAC real-time network. This connection is required to be capable of transferring block data over the network at a sustained rate of 24 million bits per second in the enhanced serial mode.

Responsiveness

One of the critical requirements for any real-time simulation system is system responsiveness. The FSCS system is required to respond to an external event, cause a short FORTRAN program to execute, and post an observable output response in less than 150 microseconds. This elapsed time, called time-critical system response, is measured at an external port on the computer. The external event occurs at a repetitive rate of 1000 events per second. In addition to the time-critical system response, CAMAC input/output response is required to be less than 200 microseconds. CAMAC input/output response is defined as the time between the action of an interrupt generated in a CAMAC crate, transfer of one CAMAC word of data, execution of the short FORTRAN program, and transfer of the CAMAC word of output.

Frame Rate

To support simulation applications needing higher frame rates, LaRC requires the system to support simulations running at 1000 frames per second. At this frame rate, during any given frame, the system must deliver at least 600 microseconds of CPU time for the simulation model with 100 bytes of real-time input and 100 bytes of real-time output. This requires that the sum of system overhead and real-time input/output be less than 400 microseconds.

Intercomputer Interface

An intercomputer interface is required to transmit

data between the simulation computers during synchronous real-time flight simulation. This facility is required to transmit data at a minimum rate of 6 megabytes per second with a maximum transport delay of 50 microseconds. The transport delay is defined as the time from initiation of transfer by a simulation program in one computer until the first byte of data is in application primary storage in the second computer.

Real-Time Data Recording and Retrieval

To support real-time data recording and retrieval during synchronous flight simulation, LaRC requires the capability to record and/or retrieve information from two files for each simulation. The aggregate storage capacity is required to be a minimum of 180 megabytes. Sufficient data rate is required so that a simulation can record or retrieve one 1000-byte record per real-time frame from each file simultaneously at a frame rate of 100 frames per second.

Language and other factors

At LaRC, almost all simulation programs are written in the FORTRAN language. Furthermore, simulations have been developed on CDC 6000 series computers and succeeding generations for over twenty years. With simulations written taking advantage of the CDC 60-bit architecture, LaRC requires that the FSCS system support simulations written in the FORTRAN language with a minimum floating point mantissa precision of 14 decimal digits and with a minimum exponent range of plus and minus 250 decimal. In addition, the C language must support a limited number of applications and Pascal is required to support the CAMAC configuration database.

An application development capability is required to operate simultaneously with simulations operating in real-time using all the real-time computing power specified. This application development capability has a minimum performance specification and requires an advanced source language level debugger.

NEW SYSTEM

The computers that LaRC is putting in place to fulfill the requirements are CONVEX Computer Corporation C200 series computers. These computers are classified as minisupercomputers and support both 64- and 32-bit scalar, vector, and parallel processing technology. The new system will be delivered in stages as the software system evolves. The first delivery consists of a CONVEX C220 (2 CPUs) with one CAMAC interface. For this publication this system is identified as the initial system. The system is delivered with two peripheral buses (PBUS): one PBUS that is used for input/output to

standard peripherals such as tape, disk, and line printer and one PBUS that is used exclusively for real-time input/output to the ARTSS CAMAC network. Each VME Input/Output Processor (VIOP) is a Motorola 68020 microcomputer that provides programmable input/output control. Each VIOP is connected to a standard 9U VMEbus and to the corresponding PBUS. The CAMAC interface consists of a KineticSystems Model 2165 Enhanced Serial Highway Driver and a VME Microsystems International Corporation VMIVME-to-DRB32E VAXBI interface VME card that provides the signal path between the VMEbus and the Serial Highway driver.

The second delivery will consist of one CONVEX C230 (3 CPUs) computer configured similar to the C220 with 2 PBUSs and one CAMAC interface. The computer will contain 256 megabytes of main memory and sufficient disk and other peripherals to support flight simulation. The next delivery will consist of a second CONVEX C230 with 2 PBUSs, one CAMAC interface, and 256 megabytes of main memory. Addition of a memory-to-memory real-time intercomputer interface will result in the configuration shown in Figure 2. Two additional purchase options will upgrade each C230 computer with an additional CPU and a CAMAC interface resulting in two CONVEX C240 (4 CPUs each) computers each having two CAMAC interfaces.

Operating System

The real-time operating system is being developed in two stages. The operating system developed in the first stage will incorporate only a real-time kernel and those other features necessary to support simulations operating in real time. With this version of the real-time operating system, application development activities and other non-real-time activities will take place on the C220 while real-time activities will execute on the first C230.

The second stage will incorporate an implementation of the full UNIX operating system using the real-time kernel developed previously. With this version of the real-time operating system, the UNIX operating system portion will be pre-empted by real-time requests and response to real-time interrupts will be deterministic and very short. This version will support all activities of a normal UNIX operating system and also simultaneously support real-time applications.

Intercomputer Interface

The memory-to-memory intercomputer interface to be delivered with the second C230 will offer a high-speed shared memory approach. A shared memory area will be allocated for intercomputer data. Any data written to the shared area on one computer will be written within a few microseconds to a shared

area on the second computer. A protocol will be defined to avoid problems with synchronization of data in the shared area.

Benefits

Engineers at LaRC use simulation in a research and development environment. This environment requires that research simulation programs be developed in a short time period and these programs are subject to frequent modification to meet the goals of the researcher. Because of these factors, there is little time available for optimization of algorithms. This requires that the computing system generate efficient execution code and provide utilities to enhance performance of the simulation models. The FSCS will provide these capabilities which allow increased performance of the CPU and hence increased simulation capability.

LaRC expects improvement in three areas critical to flight simulation: CPU power, system and input/output latency, and memory size. As mathematical models have become more complex, LaRC research programs have been restricted in the ability to run a complete model in real time. Programs are forced to run subsets of the full study because of the limitations in CPU power and memory. The simulation models continue to increase in complexity and are now requiring increased frame rates because of higher frequencies represented in the models. The improvement in CPU power and low latency will allow support of much more complex models and higher frame rates. The additional memory (unavailable on the current CYBER 175 machines) will also alleviate restrictions which require continual optimization to trade CPU utilization for memory by packing and unpacking data elements into memory words. In addition, software technology advances including tools, utilities, and languages will greatly increase programmer productivity. During this same time, real-time graphics display processors are being acquired that will contribute to increased performance of the FSCS by off-loading some of the graphics software currently required in the simulation processor.

Specific areas where LaRC expects the new FSCS to provide expanded research capability include simulation of complex models that include sophisticated avionics and weapons models that have not been feasible. Full mission simulations such as air traffic control models will be supported with much greater fidelity and will include many sub-models not possible before.

Increased performance provided by the FSCS will permit LaRC to perform simulation in new areas. Simulation of very high performance aircraft requires inclusion of the dynamics associated with the flexibility of the airframe structure. Studies will be performed that include simulation of these flexible airframe

structures. A research program investigating methods for controlling the leading edge aerodynamics of wings has been initiated. New programs involving research simulation of active control of space structures dynamics (see Figure 3) including vibration and flutter require the higher frames supported by the FSCS. Another program being initiated is research into developing better control laws for the Space Shuttle Remote Manipulating System. This program will incorporate simulation of bending modes which require higher frame rates.

A major goal of research flight simulation is to increase the fidelity of the simulation of the entire aircraft to the detail whereby the simulation performs as well as a prototype aircraft. When this goal is reached, it will significantly shorten the time required between the design and the production of aircraft. The increased modeling capacity of the FSCS will significantly contribute to this effort.

LaRC IMPLEMENTATION

The following sections discuss software and hardware items to be developed by LaRC staff. A major software effort is required to integrate the delivered systems into the flight simulation facility. A scenario describing operation of the initial system is presented to facilitate understanding of the various components being described.

Initial System Scenario

The following scenario describes the life of a single real-time job on the initial system.

The real-time simulation programmer logs on to UNIX on the CONVEX and reserves real-time resources by running the *sked* command. The user specifies the real-time CAMAC network configuration he wishes to use, the frame time, the amount of compute time needed per frame, and the amount of real-time Data Recording and Retrieval (DRR) space needed. If the resources can be scheduled, *sked* allocates the simulator sites through the configuration switch which connects the sites into a CAMAC ring network called a highway. The computer communicates with the highway through CAMAC hardware called a Serial Highway Driver (SHD). The SHD is driven by a VIOP (VMEbus I/O processor) on the CONVEX. The program that runs in the VIOP is called SHIP (Serial Highway Interface Processor).

The scheduler program *sked* reads from a configuration file the operations to be performed on modules in various CAMAC crates and sends them to SHIP. Initialization functions are performed immediately. SHIP then loads run-time highway operation lists into the SHD command memory. These lists will be used later to perform I/O in blocks for the user job when it enters real-time. Finally, *sked* creates a

symbolic link to the individual configuration file used in scheduling. The Real-time Supervisor can later access this file for information it needs such as block sizes and module types.

The user at this point has real-time resources scheduled and proceeds to prepare a program for execution in real-time. This may include text editing, compilation, or any other software functions available under UNIX. The actual real-time portion of the session takes place when the user executes an application program that, following highway input, computes the simulation model, and sends output back to the sites. To communicate with the real-time system, the user employs Real-time Supervisor which is a subroutine library that interfaces the application code and real-time environment. The real-time application makes several calls to Supervisor to initialize tables and buffers for highway and DRR I/O.

The job now enters real-time with a call to SRT (Synchronous Real-Time) or ART (Asynchronous Real-Time) which specifies the real-time state. SRT is the normal mode of real-time operation where the equations of motion are calculated frame-by-frame in synchronization with the real-time clock. I/O is allowed only to real-time devices (the highway and the DRR). ART is a hybrid state where highway and DRR I/O are allowed but operations are not synchronized with the real-time clock. I/O to normal devices is also allowed in this state.

Any errors that occur during execution cause the RTERror portion of Supervisor to gain control of the job. This routine removes the program from real-time and displays a menu to the user including options to end, abort, analyze memory, recover, or dump. The program may be recompiled, relinked, and run again.

At the end of the session, the user may explicitly invoke the real-time resource descheduler to release the VIOP, SHD, and DRR. Alternatively, real-time resources are released when the user logs out.

Initial CAMAC Software Driver

The driver software for the CAMAC highway is divided into two parts: driver code in the CPU and driver code in the VIOP. During highway initialization and non-time-critical operation, the CPU driver issues system calls to activate the VIOP portion. In time-critical SRT, the VIOP portion is always resident and polls areas in the simulation program looking for I/O requests. Since polling has much lower latency than an interrupt design, polling is the method chosen for attaining the performance needed for LaRC real-time operation.

The crucial concept in LaRC implementation of real-time simulation is the real-time frame. Frames are defined by a very accurate external device called the Real-Time Clock developed and patented by LaRC (see reference 6). This is an interval timer that

sends pulses to the site clocks (SCIUs) every 500 microseconds. The SCIU uses these pulses to define the real-time frame as an integral multiple of 500 microseconds, e.g., 1 millisecond and 12.5 milliseconds. A frame has three phases: input, computation, and output. Input and output times do not vary from frame to frame. Computation of the time-critical model and real-time output must compete prior to end of frame, otherwise an error condition is reported.

When the SCIU counter indicates that a new frame is to start, a demand message is sent upline to the SHD. SHIP detects the clock demand, inputs the real-time data, and sets a bit in the real-time program indicating start of frame. The real-time program unpacks the real-time data, does model calculations, may process asynchronous real-time I/O, and sets a flag to SHIP indicating end of model computation for the frame. The program then loops waiting for the flag from SHIP indicating the completion of input for the next frame.

During the compute phase of the real-time job, SHIP does asynchronous input/output requested by the job and monitors a timer indicating when the job's compute time for the frame is exhausted. If the job attempts to use too much compute time, the job may be aborted. But if, during initialization, the job has so requested, it will be restarted at the beginning of its compute phase rather than the point of interrupt. After synchronous output, SHIP may have time to process other asynchronous I/O requests for the job.

SHIP aborts the job for any highway status errors that occur after operations to the highway. The user job typically regains control through the Supervisor routine RTEROR and examines the type of error. The user may continue or may need to reschedule real-time resources.

Configuration Management Software

Maintenance of simulator site configuration information is done in two phases using two utilities: Site Compiler (SC) and Site Linker (SL). A simulation job typically uses one control console, a simulator site (or two), and often a site that has a graphics computer. The amount of data and the use of it almost never changes for the console. The amount of data for simulator sites can vary, especially if they contain a minicomputer.

SC and SL provide a two-step method of maintaining site configuration. SC accepts as input descriptions of the total hardware configuration of the sites and timing information. It also accepts input describing how this hardware is used, e.g., the same device could be synchronously or asynchronously used by different programs. It produces binary records for each of these means of describing the various sites. A total SC run is needed relatively infrequently such as when a new site is added.

More commonly a user may want to enhance a simulation by adding another site to it e.g., a graphics minicomputer. SL, the site linker, would be used for this operation and would merely read the appropriate records built previously by SC and link them together into one file. Input to SL is simply a list of site files to be linked together and the name to give to the resultant configuration file.

Real-Time Supervisor

Real-time Supervisor is a suite of routines written in C, FORTRAN, and assembler that is contained in a run-time library used by the application programs. It supports most of the job's interactions with the system and the real-time environment. Supervisor has three main classes of functions. First, it interacts with the sites on the highway. Second, it drives the real-time recording device (DRR). Third, it does terminal interaction and error processing.

The highway portions define communication areas and buffers, make asynchronous requests for input or output, retrieve and reformat this data, process and reformat the synchronous I/O and fetch up-line quasi-interrupts called demand messages.

The DRR portion defines buffers and variable names for recording data, performs input and output operations and repositions the DRR files.

The terminal I/O and error recovery portions are the most complicated. One of the main aspects of the real-time design philosophy is that the application does not terminate unless the user tells it to. The RTEROR routine recovers from every error it can and displays a menu giving the user several options. The user can recover if a predefined recovery address has been specified, and may display and analyze portions of code and data, and may dump memory (and even restart the memory image later).

Two standalone utilities are related to Supervisor. These are RTGO, which restarts partially executed binary images, and DP, an editor/analyzer of memory images (or any binary file).

ARTSS Hardware Modifications

Some modifications to the ARTSS are required to support the new system as well as to support new requirements. The CAMAC configuration switch controller requires expansion of the number of RS-232 ports during the transition period. The switch controller will be replaced by a UNIX-based microcomputer. This increases the maintainability of this hardware element. With the requirement of enclosing one simulation computer in a secure room, an additional clock is required. With the capability of supporting simulations at higher frame rates, a smaller basic time interval is required from which the simulation frame time is derived. A new real-time clock is

being built and will be housed in the secure facility. The original clock will be modified to supply a shorter basic time interval.

Other Software

The current system has the capability of downloading load map and symbol table information into the simulation programmer's console. This information is used to address symbolically data in the CYBER 175 mainframe for display and modification from the programmers console. Generation of software on the CONVEX computers and modification of software in the programmers consoles will be required to support this valuable feature.

In addition to supporting time-critical synchronous simulation, a facility called background processing is supported. With this facility, a simulation application identifies a portion of the program that is the time-critical section and a second section called background that is not as critical but still must be done during synchronous real-time. During any given frame, the time-critical section is done first and time left over is given to the background section. The background section is allowed to execute until the beginning of the next frame where it is interrupted and the time-critical section is executed. This continues with the background portion spanning frames until it finally completes (or never does as the case may be). The capability of interrupting the background section and allowing it to resume later on a subsequent frame will be supported. The CONVEX implementation of UNIX involves stacks and the calling of subroutines causes addition and deletion of stack information. This may pose a problem with the LaRC implementation of background processing.

The current real-time operating system on the CYBER 175 computers supports a high degree of recoverability for real-time jobs. The operating system has been modified so that most usually fatal errors are recoverable to the Supervisor in the real-time job. Following error recovery the simulation operator is given several options for further processing. Similar capability is required for the new system.

Data Formatting Issues

Two issues have arisen concerning data transfer between the host (CONVEX) computers and other devices connected via the ARTSS CAMAC system. The first issue concerns data format as it passes through the Serial Highway driver. Data is transformed as it passes through the device. The driver operates in either 24-bit or 16-bit mode. In 24-bit mode the driver reads in a 32-bit word, but only transmits the lower 24 bits to the highway. This results in having to reformat the output into 4 byte packets with the high order byte containing filler.

When operating in 16-bit mode, the driver transmits all bytes read from memory. However, it inverts the high order 16 bits with the low order 16 bits. Hence, if the CONVEX memory contains the string "abcd", it is received downline as "cdab". Thus data formatting increases latency of transfer because software must be employed to manipulate the data before or after transfer. In addition, the 16-bit format decreases the effective bandwidth of the CAMAC highway.

The second issue concerns binary data transmitted from the host computer to another device. In most instances, when a new device is added to the network, the format of binary data transmitted over the highway is determined by the characteristics of the device including the software available on that device. Often, the host computer reformats data for consumption by the downline device. This data manipulation steals computation time from the simulation model. Alternative solutions include: (1) forcing the downline device to convert host data formats, (2) developing a common data interchange format that consumes little conversion time, or (3) adopting the IEEE standard for all binary data transfers.

STATUS AND PLANS

As of this writing, the initial C220 system has been delivered and is in acceptance testing. Preliminary results from acceptance testing show that the system meets all the performance criteria as shown below:

Test	Requirement	Measured
CPU Performance	<330 secs	317 secs
Time-critical system response	≤150 μs	31 μs
CAMAC input/output response	≤200 μs	152 μs
CPU time available at 1000 frames per second	≥600 μs	700 μs

Implementation of real-time Supervisor and configuration management software were begun in late 1989. The design for SHIP, the serial highway driver software, was begun following design reviews of the operating system in the spring of 1990 and implementation is beginning. The first version of the system software will support a fixed site configuration that is capable of simulating two aircraft in a dual-dome simulation. A target application has been selected. Integration of the system software and conversion and integration of the application into an operational simulation is expected by the end of the year.

Following testing and verification of the initial simulation, general purpose configuration manage-

ment software with a non-configuration dependent SHIP will be integrated to support any arbitrary combination of simulation sites. At this point the system software will support any existing application. This is planned for the spring of 1991.

With delivery of the first phase of the final operating system, the LaRC system software will be modified to support the new real-time kernel. Multiple simulations per computer will be verified as well as multiple CPUs per simulation. This work is planned for completion by late summer 1991. With delivery of the final phase of the operating system, the initial LaRC software will be phased out, and final software to support the general UNIX environment will be generated. The final software is expected to be completed in the spring of 1992.

CONCLUSION

To meet the requirements of NASA Langley Research Center to simulate the increased complexity and higher performance of modern aircraft, a flight simulation computing system with very high scalar performance is required. A solution using centralized minisupercomputers coupled with a proven real-time network technology will provide research scientists and engineers with the tools necessary for high performance flight simulation.

REFERENCES

1. Crawford, D. J. and Cleveland, J. I. II, "The New Langley Research Center Advanced Real-Time Simulation (ARTS) System," AIAA Paper 86-2680, October 1986.
2. Crawford, D. J. and Cleveland, J. I. II, "The Langley Advanced Real-Time Simulation (ARTS) System," AIAA Journal of Aircraft, Vol 25, No. 2, February 1988, pp. 170-177.
3. Crawford, D. J., Cleveland, J. I. II, and Staib, R. O., "The Langley Advanced Real-Time Simulation (ARTS) System Status Report," AIAA Paper 88-4595-CP, September 1988.
4. Cleary, R. T., "Enhanced CAMAC Serial Highway System," presented at the IEEE Nuclear Science Symposium, San Francisco, California, October 23-25, 1985.
5. ANSI/IEEE Standards 583, 595, and 675, Institute of Electrical and Electronic Engineers, 1976.
6. Bennington, D. R., "Real-Time Simulation Clock," LAR-13615, NASA Tech Briefs, June 1987.

BIOGRAPHY

Jeff I. Cleveland II is the Project Engineer of the LaRC Flight Simulation Computing System. He received a B.S. in Electrical Engineering in 1963 from Texas A&I University and an M.S. in Electrical Engineering and Computer Science in 1970 from The George Washington University. He has worked in the field of flight simulation and operating system software since 1963.

Steven J. Sudik is a Project Analyst with Unisys Corporation. He is the group leader for the Flight Simulation Computing System software development team. He received an A.B. in English and Philosophy from Marquette University in 1966 and an M.A. in Philosophy from Indiana University in 1971. Prior to coming to LaRC he held positions as a systems programmer and business applications programmer. He has worked at LaRC since 1975 in operating systems support.

Daniel J. Crawford was the Project Manager of the LaRC ARTSS Project. He received a B.S. in Physics in 1961 from the University of Massachusetts at Amherst and an M.S. in Electrical Engineering in 1975 from The George Washington University. He has worked in the field of flight simulation since 1962.

TRADEMARKS

UNIX is a trademark of American Telephone and Telegraph.

CDC and CYBER are trademarks of Control Data Corporation.

CONVEX is a trademark of CONVEX Computer Corporation.

DEC, VAX, and VAXBI are trademarks of Digital Equipment Corporation.

ELXSI is a trademark of the ELXSI Corporation.

Concept, Gould, and NP2 are trademarks of Encore Computer Corporation.

CT-6 is a trademark of Evans and Sutherland.

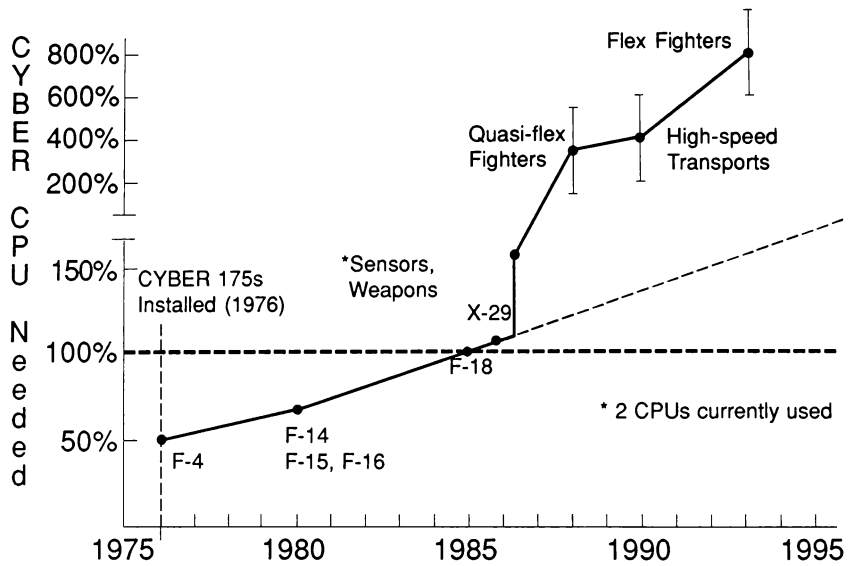


Figure 1
CPU Requirements to Support Flight Simulation

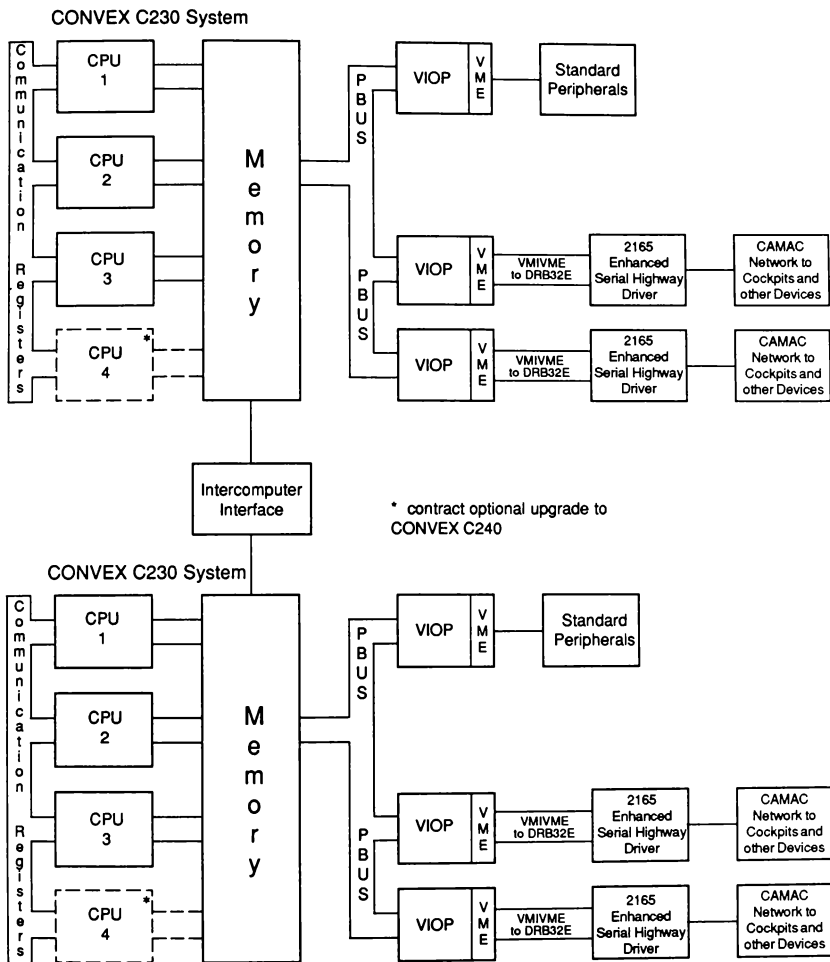


Figure 2
Final System Configuration

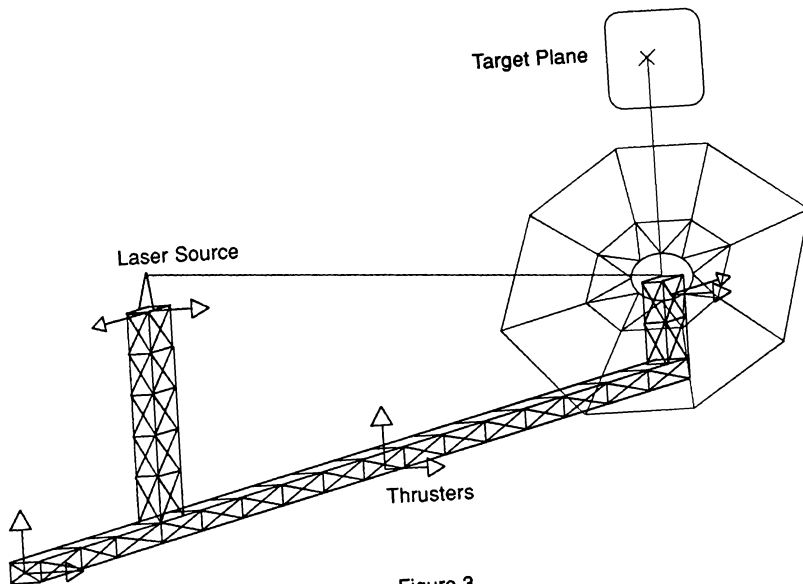


Figure 3
Space Structure with Active Control

USE OF FLIGHT SIMULATION ACCELERATORS
FOR ENHANCED HIGH SPEED COMPUTATIONAL CAPABILITY
IN REAL-TIME MANNED FLIGHT SIMULATION

Patrick K. Moriarty*
Craig A. Wilsey*
Jan L. Wurts*
Lockheed Aeronautical Systems Company
Burbank, California

Roy B. Hollstien**
David S. Hollstien***
RDH Simulation
Paso Robles, California

Abstract

This paper discusses the development of current applications, and the proposed future applications, of the RDH Simulation Flight Simulation Accelerator® (FSA) boards used at Lockheed's Weapon System Simulation Center (WSSC) at Rye Canyon.

The focus of this paper will be the two primary current applications of the FSA to real-time manned flight simulation, the P-7A handling qualities simulation, and the real-time aircraft models used in full mission simulation, with emphasis on the net improvements of these configurations with use of the FSA. Special programming considerations made for the FSA during software development, as well as methods for FSA software structure and validation will be included.

Introduction

In late 1987, WSSC had a requirement that, in part, mandated the real-time manipulation of large data bases for function generation purposes. Table look-up function generation had previously been performed, in real-time, on array processors. But with the advent of databases defined by up to 544 separate, mostly bivariate and trivariate,

data tables, 165K function data points, and the need to drive the table look-ups at frame rates of 100 hz or better, the need for a new methodology became clear.

Matching complementary needs and capabilities, WSSC and RDH Simulation teamed together to their mutual benefit. RDH Simulation provided a potential solution to WSSC's function generation dilemma in the form of a prototype hardware device designed to perform high speed linear interpolations of arbitrary functions, that was compatible with WSSC's Gould CONCEPT/32 computers. WSSC provided RDH with access to a real-time simulation where the capabilities of the FSA could be fully exploited and verified. In addition, the FSA's supporting software could be field-tested, debugged, and developed further.

As a preliminary experiment, FSA was interfaced with the Gould via a standard Gould HSD card, and contained 64K (32 bit) words of common memory for I/O, 64K words of memory for breakpoint tables, and 256K words of memory for function tables. Function and breakpoint data were downloaded to the FSA prior to the start of the simulation. During real-time simulation, at the beginning of each frame, an input buffer made up of the vehicle state data required for the table look-ups was downloaded to the FSA. The FSA

*Research Specialist Sr., Member AIAA

+Research Engineer

**Research Engineer

++Partner, RDH Simulation

* * *Partner, RDH Simulation

Copyright © 1990 by Lockheed Corp.
Published by the American Institute Of
Aeronautics and Astronautics, Inc. with
permission.

would then process the function interpolations, in parallel, while the Gould continued processing code. The function generation results were then uploaded to the Gould just prior to when the results were needed (Figure 1).

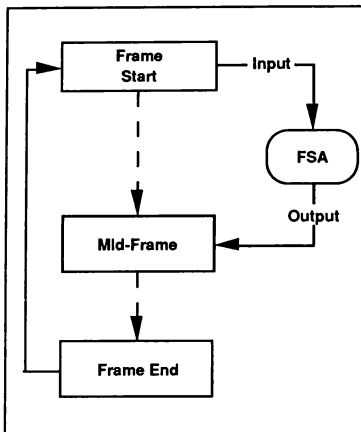


Figure 1. FSA/Simulation Interface

Using the FSA solely for function generation produced results that far exceeded WSSC's expectations. The FSA not only was able to handle the larger data bases at the required frame rates, but actually lent to an increase in the amount of available background time per frame. Using the experience gained by the experiment, it was realized that with the proper supporting software, and with no modification to the hardware, the capabilities of the FSA could be expanded far beyond the function generation tasks for which it was originally designed. These increased capabilities, and the way WSSC exploited them for its current programs provides the basis for the balance of this paper.

The RDH Flight Simulation Accelerator

The FSA is a programmable processor that is architecturally designed for rapid execution of flight simulation computing tasks such as function generation, aerodynamic coefficient build-up, vehicle dynamics,

engines, flight controls, avionics, etc., in parallel operation with a host computer or other processors. To enhance its real-time aircraft handling qualities and tactical mission simulators, WSSC has installed FSAs in Encore CONCEPT 32/9780 and MultiSel 32/6745 computers. These FSAs are single, 15 by 18.5 inch printed circuit boards that mount in SelBUS slots of the host machines, with connection by two flat cables to Encore 9131 high speed data (HSD) interfaces.

Up to eight FSAs may be daisy-chained to one HSD, or to RDH Device Communication Adapters (DCA) that facilitate direct data transfer between FSAs, or FSAs and HSD compatible I/O devices (Figure 2). There are no additional power or floorspace requirements for FSA installation in Encore CONCEPT 32 machines.

Programs executed by the FSA are edited, compiled, and stored on disk similar to programs that run on the host; they are written in a subset of FORTRAN 77+ with macro extensions that simplify real-time flight simulation by providing:

- Built in atmospheric functions for sonic velocity, air density, etc., versus altitude;
- Single and double precision accumulation in Euler, Tustin, or Adams-Bashforth integration algorithms;
- Euler angle wrap-around integration without gimbal lock, coordinate transformation using Euler angles or quaternions;
- Latitude and longitude integration with flat earth velocity inputs and double precision output for visual systems; and
- Control element models with rate and output limiting: integrator, first and second order lags, first and second order lead-lag, washout, etc.

The macro library can be modified or extended to satisfy special flight simulation requirements.

FSA programs have the form of FORTRAN subroutines without formal arguments. Values are passed from the host to two input common blocks in the FSA interface memory allocated to a subroutine; one block stores constants or parameters that remain fixed

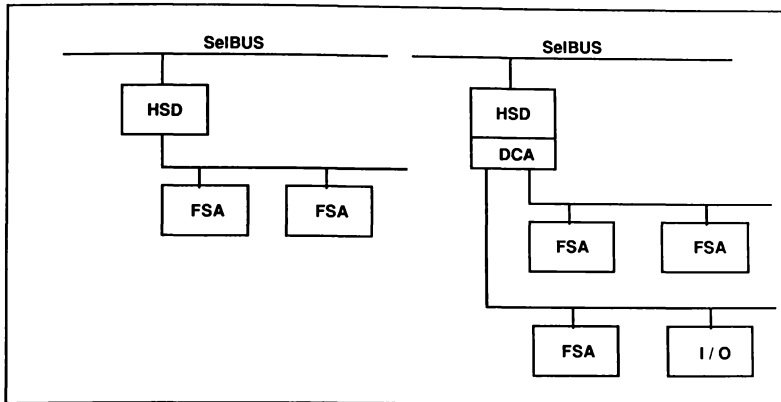


Figure 2. FSA/Host Interface Architecture.

during real-time operations, and the other is updated before each execution. Values are returned to the host from an output common block after each subroutine completion. A correspondence between host arrays and FSA subroutine input and output common blocks is established during the host program initialization, and the data are automatically transferred between them during the FSA subroutine initialization and execution (Figure 3).

Although transparent to the user, external functions in FSA FORTRAN programs are evaluated by FSA hardware elements with:

- Actual 15 Mflops performance on function generation applications;
- Function grouping to eliminate redundant breakpoint search;
- Up to eight independent variables;
- Optional "Step" and "Binary" breakpoint search methods;
- Multiple function invocations without duplication of data;
- Optional endpoint "truncation" or "extrapolation" rules; and
- Separate library files for convenient and centralized function

data management.

In the example below, functions CLA and ZMA, defined over the same breakpoints, are "grouped" for evaluation using one breakpoint search; their values, intermediate results K1, K2 ..., and "non-grouped" functions ZNB, ZNDA, ZNP..., may be used in arithmetic expressions for computation of total aerodynamic coefficients CLTOT...CNTOT.

```

...
CLA = CLA(ALPHA,DELFL)
ZMA = ZMA()

R2VT = 0.5/VT
K1 = CBAR*R2VT
K2 = BREF*R2VT
CLTOT = CLA + FGE*DCLIG + CLDE +
&CLDH*DELH + CLSPL + CLSPR
&+ K1*(CLQ*Q + CLAD*DALPHA)
&+GEAR*CLGR

CNTOT = BETA*ZNB(ALPHA,DELFL) +
& F*ZNDA(DELA,ALPHA)
&+ ZNDR*DELR + ZNDRT +
& ZNSPL + ZNSPR
&+ K2*(ZNP(ALPHA)*PS + ZNR*RS)
...

```

FSA FORTRAN subroutines are translated into executable modules by a cross compiler, assembler, linker, and load module generator. The transformation may be performed in

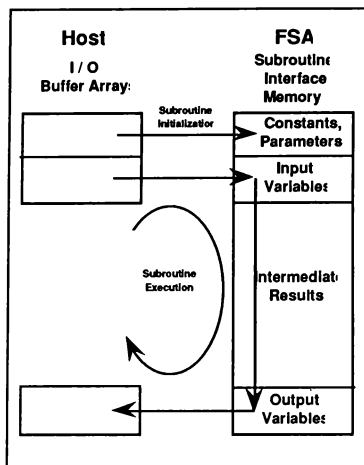


Figure 3.
FSA/Host Software Interface.

separate steps, or by one MPX-32 operating system TSM macro. Separately compiled subroutines are relocatable in the FSA, and (within the limits of available memory) any number of them may be downloaded for execution as required by CALL statements in host programs.

Routines that can be executed concurrently may run in the host and FSA, or in multiple FSAs. Variables computed by one routine that are needed by the host program and as inputs to another routine are passed through the host's interface buffer arrays. Variables computed in the FSA may also be passed by global address reference to other subroutines in the same FSA.

Control of real-time FSA operation is performed by FORTRAN callable routines that:

- Attach and detach HSD units and connected FSA boards;
- Transfer load modules and verify their data;
- Set initial condition values of state variables;
- Set the integration time step;

- Transfer subroutine parameter values, subroutine input variables, and subroutine output variables;
- Start FSA subroutine execution;
- Synchronize execution of host programs and FSA subroutines;
- Read HSD and FSA status and error conditions; and
- Zero FSA memories.

FSA programs are tested by running them under the control of an interactive debugger. From a host computer terminal, users can:

- Load any number of executable modules into an FSA;
- Verify loaded modules by reading them back for comparison with disk files;
- Set initial conditions of state variables and values of subroutine input variables from prepared files, or from the keyboard;
- Control subroutine module execution;
- Display values of all subroutine output variables;
- Display individual values of inputs, outputs, and intermediate results by symbolic reference to their source code names;
- Symbolically reference and show function data, breakpoint values, input variable names, and interpolation results of all user defined function generation operations; and
- Compare automatically the current subroutine output variable values with values saved earlier in check files.

Current WSSC Applications Using the FSA

P-7A Handling Qualities Simulation

Aircraft models used for flight controls development, handling qualities evaluations,

and other high gain tasks, are complex. It is common to have the comprehensive force and moment buildups, which involve the aerodynamics, landing gear, and the propulsion system, be a function of the physical configuration characteristics, such as control surface deflection, angle of attack, gear location, thrust vectoring, and engine dynamics.

Requirements of this type of aircraft model are that it be an accurate, validated model. It also must be flexible enough to accommodate rapid changes to any of the aerodynamics, propulsion, landing gear, flight controls, hinge moments, or actuator components of the model. Since these components, along with the equations of motion, atmosphere, navigation, autopilot, and data acquisition are the basic components of the simulator, modularity becomes an inherent requirement.

When the development of the P-7A handling qualities simulator began in October 1988, the job was driven not only by the above requirements, but also by strict budgetary requirements. Therefore, the simulation was to reside in a Gould MultiSel computer. This computer is the less expensive, smaller capacity version of the dual processor 9780, and has only 2 to 3 MIPS computational speed.

To meet the demanding requirements with limited capacity, a FSA was linked to the MultiSel via a HSD connection. The initial intent was to use the FSA only for function generation. However, by using the FSA FORTRAN software capability, a new simulation architecture was established and embedded into the FSA's memory. This new architecture, which included a software programming capability and macro extensions, provided the ability to use the FSA not only as a function generator, but also as a third parallel processor for logic and floating point calculations. As more of the basic components of the simulation were off-loaded into the FSA, more CPU/IPU task space became available for software expansion on the Gould MultiSel.

The current software division of CPU, IPU, and FSA responsibility for the P7-A has three tasks in the FSA running in parallel with the remaining CPU and IPU tasks on the MultiSel (Figure 4). Task 1 computes the aerodynamic coefficients and the control surface hinge moments in aircraft stability axes. Task 2 converts the aerodynamic coefficients into body axis coordinates, and uses these body axis force and moment coefficients to compute body attitude as well as angular and linear body rates, and

accelerations.

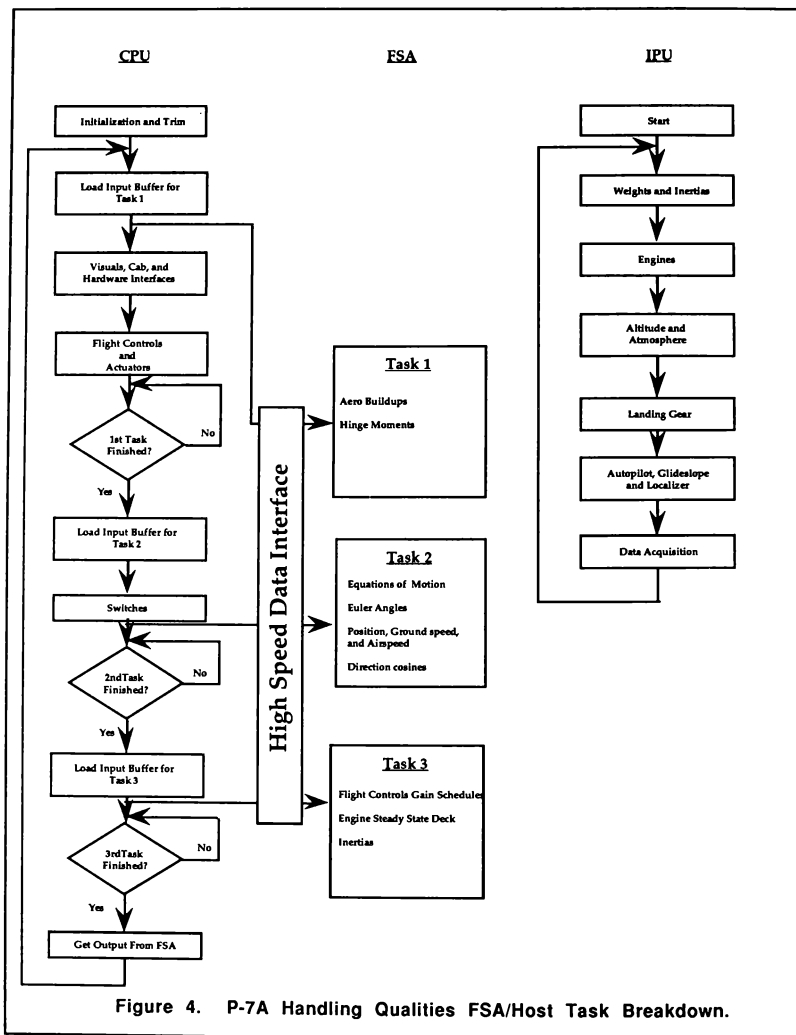
Task 2 also computes Euler Angles using second order Adams Bashforth integration, ground speed, airspeed, geocentric positions, and current direction cosines. Task 3 performs the function generations for the flight controls gain schedules, the propulsion steady state values, and any additional minor function generations not applicable to Tasks 1 or 2.

The FSA has been used extensively to conserve limited CPU task space. CPU task space is the critical element to the success of the P7-A simulator, since only the CPU can contain I/O related software drivers. The simulation will soon be required to tie into an iron bird, and an avionics system integration laboratory with flight hardware in the loop via 1553 and ARINC 429 buses, and both will require additional task space in the CPU to assemble, transmit, and receive the bus messages through DATAPool.

An asset of the FSA is that it is programmed in code which is readable and similar to the FORTRAN 77+ used in WSSC simulation programming. FSA FORTRAN has actually been unit tested for syntax errors with a FORTRAN 77+ compiler and can be easily and rapidly changed to accommodate reconfigurations in the model. Prior use of array processors for embedded programming was slow and involved considerable knowledge and experience with the array processor machine-level languages.

While programming the FSA, special attention was paid to the modularity of the embedded software. The resulting tasks were created with the ability to replace any single module without having to recreate or modify the others. In conjunction with the multiple Function Data Libraries (FDL's), this modular architecture provides the simulation programmers with a means of rapidly understanding and reconfiguring the simulator's software.

The P7-A handling qualities simulator is currently using a 62.5 Hz (16 msec) frame rate. Although a higher frame rate has also been achieved comfortably, since the P7-A is not a high frequency airframe, simulation time frame size is not an overriding priority and the increased fidelity from smaller frame times does not justify giving up the extra background processing time in the MultiSel. Several validations have shown that the 50 to 64 Hz full six degree-of-freedom models on the simulator can very accurately match the closed loop dynamics generated by much higher fidelity non-linear analysis models like ACSL or CSMP. In subsystem models where a higher



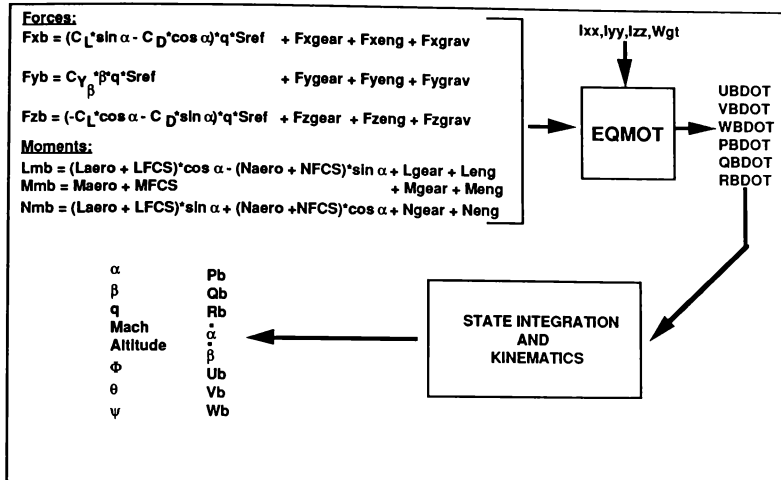


Figure 5. Full-Mission Simulation Aircraft Model.

level of fidelity is desirable, such as control surface actuators, or landing gear models, multiple internal frame loops can usually improve the frequency response and phase margin substantially.

Several confidence-level timing tests were run on the simulator's CPU resident subroutines using the system timer. These tests indicated the collective times of all the CPU subroutines were in the range of 8 to 12 msec for a 64 Hz model fully configured with flight controls, engines, and cockpit hardware interfaces. As subsystems were removed from the IPU and down-loaded into the FSA, not only was space made available for future software expansion on the IPU, but also the collective times for the CPU tasks showed no increase indicating the CPU was never waiting for the FSA to complete any of its tasks.

Airframe Models Used for Full-Mission Simulation

In order to provide a high fidelity detailed battle environment for full-mission simulation, multiple aircraft models, of varying types, must be provided in a real-time environment. Flexibility and a highly modular design were primary drivers

in providing for a rapidly changing threat environment, and enable rapid set-ups of new scenarios.

While aircraft models used for flight controls development, handling qualities evaluations, and other high gain tasks are, by nature, complex, models used in a full-mission simulation environment cannot afford the luxury of modeling on a sub-system level, but rather, directly model the performance and handling characteristics of a fully augmented aircraft (Figure 5). Each model is carefully "tuned" to match each aircraft's performance in areas of specific excess power (P_s), maximum sustained load factor (N_z), maximum sustained roll rate, etc. and handling characteristics such as short period frequency and damping, dutch roll frequency and damping, and maximum roll rates and roll mode frequencies.

Prior to the start of simulation, the proper mix of aircraft numbers and types are downloaded to the FSA. The FSA maintains each models' base address, therefore each model is completely relocatable, and can be symbolically addressed independently from the other aircraft models. Each 256K FSA board can hold about 24 high fidelity aircraft models of the type used at WSSC for full-

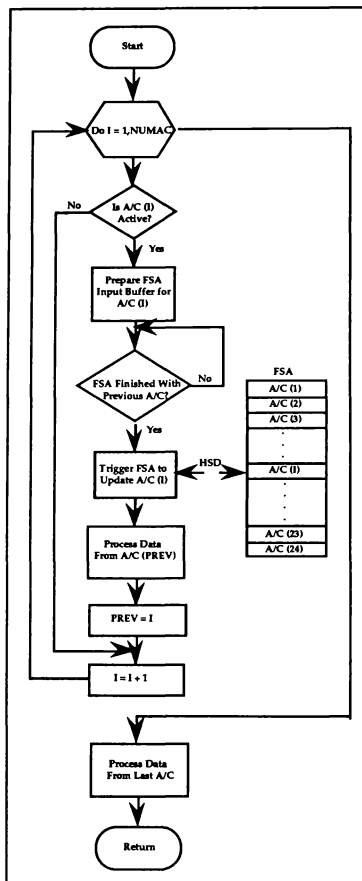


Figure 6. Full-Mission Simulation, Host-FSA Software Interface.

mission simulation. Each aircraft model contains independent force and moment build-ups, flight controls, kinematics, equations of motion, and integrates the calculated derivatives to update the vehicle state.

While the simulation is running, only the models currently active in the scenario are updated in order to minimize the overhead inherent in HSD data transfers. While the FSA is updating an aircraft, the Gould host is processing the output from the previously updated aircraft, and preparing the FSA input buffer for the next aircraft to be updated (Figure 6).

By interleaving the vehicle update tasks, resident on the FSA, with the Gould's postprocessing tasks of the previously updated vehicle and the input preparation for the next vehicle to be updated, up to 16 aircraft have been flown, at one time, with a 60 hz frame rate.

Future Plans for the FSA at WSSC

With the FSA rapidly evolving into a parallel, multi-functional processor, new applications are being explored that can best exploit the FSA. The best candidate applications are those that involve repetitive loops through the same "boilerplate" code with a different set of conditions. Candidates currently under study for FSA application include:

- Relative geometry which determine the orientation of a body with respect to other bodies (elevation, azimuth, bearing, etc.). This application is particularly well suited since the FSA's intrinsic trigonometric functions are very fast.
- Multiple avionics and sensor models for full mission simulation. By exploiting the inherent modularity that exists with relocatable FSA load modules, different avionics and sensor models can be mixed and matched with different airframe models.
- Providing full handling qualities fidelity aircraft models for the tactical full mission simulator. Using one or more FSAs in parallel, aircraft models can be provided with modelling at the sub-system level without further taxing the resources supporting the battle environment.

A. D. Patel and D. W. Baltrus
Taurus Technologies Incorporated
Newport News, Virginia

Abstract

Current advances in distributed computing, especially the "scalable" multiprocessor architectures, offer the potential for economical solutions to real-time computing applications. Unfortunately, raw computing power alone does not meet the demands of deterministic applications such as real-time simulation and control. The need for concurrent data sharing in multi-tasking, multiprocessor systems is the bottleneck in utilizing these technologies in real-time applications.

The approach presented here is an actual implementation of a virtual shared memory Interconnect (VSI). The design objectives in this project were to minimize data latency between processes and provide process transparent communication requiring shared data and cross-coupled interrupts between CPU's. The realization of these goals affords attainment of maximum available CPU band width and brings a truly scalable architecture to the real-time arena.

Introduction

Advances in computer hardware and software architectures in the past few years have significantly changed the structure of the general purpose computing industry, and in particular, the cost and distribution of computer resources. The growth in numbers and capabilities of the PC, workstation, and networking market segments has created a new technology base, and a realization of truly distributed computing. Likewise, the recent innovations in microprocessor-based RISC architectures have brought mainframe computing power to the desktop.

These advances, while radically changing the general purpose computing

market, are only beginning to be seen in the real-time market. Thus far, real-time distributed systems have been relying on raw computing power to satisfy the these needs. The current "scalable" distributed processing systems, applied to real-time computing problems, have not yet taken advantage of the cost and performance of microprocessor technology available today. The primary reasons for this are the classic problems associated with concurrent shared memory and determinism in multiprocessor systems real-time applications.

Problem

The issue of shared memory resources has been identified as a fundamental weakness in current distributed real-time systems for some time. The mechanisms which provide for interprocessor communication and synchronization rely heavily on shared memory resources. These mechanisms are critical to both performance and fidelity in complex simulation modeling applications. Typically these mechanisms either add to the processor overhead or introduce non-deterministic behavior into the simulation.

In a typical frame-oriented process simulation model, each physical system model must communicate data to other system models and input/output at precisely defined frame boundaries. In designing a distributed multi-processor system to meet such constraints, the underlying architecture must accommodate this function at the lowest possible level of hardware and software implementation to minimize the overhead involved in implementing such a simulation model.

The shared memory performance problem can be broken down into two aspects: data latency and accessibility. Both of these must be optimized to maintain

system level determinism. The basic communication mechanisms which support shared memory must accomplish these functions as a priority to avoid problems introduced when an existing architecture is adapted to real-time applications. For this reason, the real-time community has not been fully able to share in the cost advantages of distributed processing with today's high-performance, RISC-based microprocessors.

Solution

The Virtual Shared Memory Interconnect (VSI) is a solution to these problems faced by real-time computing.

The VSI is an independent bus which enables the close coupling of individual processors for high-performance, transparent multiprocessing applications. The VSibus is defined as a data-only bus separated from the local processor and I/O buses, but mapped as a direct local resource to each node. The VSI is a transparent link of real-time functions, such as interrupts and semaphores, between the individual memory subsystem components in each of the processor nodes (in a multiprocessor system), eliminating the usual memory access delays of typical shared memory approaches.

The VSI allows up to 16 processors to access the same data simultaneously. This design allows the system to be expanded to include 240 processors by allowing up to 16 clusters of 15 processors each to share the same data simultaneously. The addition of processors to the system results in linear increase in performance as there is no overhead on the processor itself, or on other processors in the system.

In the traditional shared memory approach, shared data is resident at a location that can be accessed by all processors in a highly restrictive manner. In the VSI approach, shared data is resident at multiple locations simultaneously. The update of data by any processor is shared in a deterministic manner with all the processing elements in the system. The VSI is extremely flexible in its functionality by allowing all or any of the techniques of data sharing: unidirectional, bidirectional, pro-

cessor-to-processor. It is also different from the large message-passing schemes which have large inherent overhead. The VSI is transparent to the processors, thus it does not affect their performance. The VSI also allows the system to perform multitasking among the multiprocessor systems.

The VSI, by design, is a data-oriented bus as opposed to being a control and I/O-oriented bus. All cycles on the VSI are data cycles.

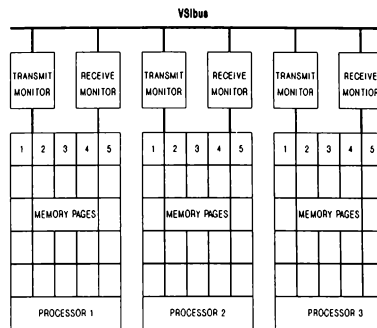


Fig. 1. Data Cycles

As shown in the figure, each processor node has a receive and transmit monitor which links its memory to the VSibus. The transmit monitor stores information on which pages of memory its processor shares with other processors on the system. The receive/transmit monitors ensure that there is no overhead to the processor by independently buffering memory updates. This allows the VSI to transfer data at 80 MB per second using current technology. By comparison, the VMEbus has a peak transfer rate of 20 MB per second, and an average of 8 to 10 MB per second performances. As all cycles are data cycles on the VSibus, its average and peak performance are one and the same.

The VSI has an inherent fairness algorithm incorporated into its bus logic which allows all processors on the bus an equal chance to transmit in a given period of

time. This allows deterministic update of information along the bus.

The VSibus has 64-bit data lines which allow it to be very effective with today's high-performance processors which are being designed with 64-bit architecture. In addition, the VSI allows for a heterogeneous computing environment. This implies that different processors and families of processors can be used in the same system. This design feature makes it possible to upgrade system performance by incorporating new processors into the system as and when they become available, without any additional cost.

Implementation

The first implementation of the VSibus has been completed on Taurus' V860 system. The V860 system is designed to provide the VSI capability while also providing for economical hardware and software compatibility. A V860 system is comprised of 1 to 240 V860 processor boards, each having an Intel i860 CPU operating at 33 or 40 Mhz, 8 to 32 MB of private memory, and 1 to 8 MB of shared memory with both the VSibus and VMEbus. The VMEbus was chosen as an auxiliary I/O bus for the processor. In addition to the processor access to the VSibus, writes from the VMEbus are also broadcast on the VSibus, thus providing a shared memory resource on multiple VME applications such as intensive I/O applications.

The local shared memory on each node is addressable from the local processor (an Intel i860), the VMEbus, or the VSibus. Addressing is virtual and each 8 KB block memory can be mapped at any address on the VSibus dynamically. VME address mapping is contiguous with a programmable base address. Each processor board includes additional local (private) memory for program execution and non-shared memory. The architecture is such that the i860 CPU can run totally isolated from the shared memory to maintain cache coherency and access the shared memory only when required.

The Taurus V860 system uses the VMEbus as an I/O path to take advantage of the wide range of off-the-shelf I/O boards

available. Since the VMEbus is totally asynchronous from the VSibus, an independent VMEbus can be generated at each node as required. Likewise, the LynxOS real-time operating system was chosen to provide an open standard platform for software development. Real-time support of the VSI features of the V860 have been made at the kernel level using a Taurus-adapted version of Intel's IAPX kernel.

The V860 board was designed with the capability of hosting CPU's other than the Intel i860, thus providing the data transparency of the VSibus to support applications demanding binary compatibility with other CPU's.

Summary

The architecture of the Virtual Shared Memory Interconnect, as proposed and implemented, provides an economic means of realizing a fully transparent and fully functional shared memory implementation for real-time deterministic simulation applications. The implementation maintains an open systems environment to accommodate the rapidly evolving processor technology.

5"-SIZE LIQUID CRYSTAL FLAT PANEL DISPLAY EVALUATION TEST BY FLIGHT SIMULATOR

Hiroyasu KAWAHARA*, Akira WATANABE*, Kaoru WAKAIRO*
National Aerospace Laboratory, Japan

Tomoyuki UDAGAWA**, Yoichiro KURIHARA**, Kengo TAKEDA**
Yokogawa Electric Corporation, Japan

Abstract

In recent years, color liquid crystal displays are being developed as next generation-indicators of flight instruments. National Aerospace Laboratory and Yokogawa Electric Corporation held a evaluation test on the function, performance, display format, etc., of 5 x 5 inches-size flat panel display. This evaluation test was conducted by a total 23 evaluation pilots on flight simulator. We confirmed that the flat panel display was effective as indicator of flight instrument. This paper describes the result of this evaluation test.

1. Introduction

In recent years, electronic flight instruments employing CRT (Cathode-Ray Tube) displays have been installed in large airplanes to reduce the work load of pilots, to attempt integration of information and to improve readability, together with increase in performances and size of aircraft. In addition, the electronic flight instruments system is so mechanized that a failure of a system component can be backed up by remaining system components.

At present, the CRT display is mainly used as an electronic flight instrument, and flat panel displays (hereafter called, "FPD") are being developed as next generation-indicators. Some of the FPD's have already proceeded to the practical stage. In this case, the FPD employs a color liquid crystal panel excellent in visibility, compact and lightweight respects.

Yokogawa (Yokogawa Electric Corporation) is developing a 5 x 5 inch-FPD unit for avionic use. NAL(National Aerospace Laboratory) and Yokogawa held on evaluation test on the function, performance, display format, etc., of this FPD unit. This evaluation test was conducted as coopation on the NAL's flight simulator.¹⁾

First we will hereunder describe the outline of a liquid crystal display (hereafter called, "LCD"), and next we will report the contents and results of the flight simulator evaluation test.

2. LCD Panel

Depending on the operation of liquid crystals, the LCD is available in two kinds, active-matrix TFT-TN (Thin Film Transistor-Twist Nematic) type or not. A TFT-TN type LCD was used in this evaluation test, because it has high contrast and is excellent in respect of response properties.

2.1 Structure and Operation of LCD Panel ²⁾

The LCD panel consists of the following items.

- (1) Liquid crystal panel
- (2) Color filter
- (3) Two polarizers
- (4) Back light

Figures-1 and -2 show an outline structural drawing and the principle of operation of the LCD panel, respectively.

Liquid crystal molecules are rotated optically by 90° in the LCD panel, because usually, such liquid crystal molecules are twisted spirally by 90° and input light goes along them. The LCD panel is put between two polarizers which cross at an angle of 90°, and when voltage are not applied to the TFT (Thin Film Transistor), light inputted from the back light will pass through the LCD panel without any change. Reversely, when voltages are applied, the light will be shut with the polarizers and will not be able to pass through the LCD panel, because liquid crystal molecules will have no twist. Color images can be displayed by controlling the TFT, because RGB (Red, Green, Blue) color filters are arranged corresponding to every picture element of the LCD panel.

2.2 Features of LCD

As compared with the CRT display unit, the LCD display unit possesses the features as shown in Table-1. Table-2 shows performance of the LCD panel³⁾ used in this evaluation test.

* Flight Simulation Lab., Control Systems Div.

** Section 2 Engineering Department Aerospace & Marine Products Div.

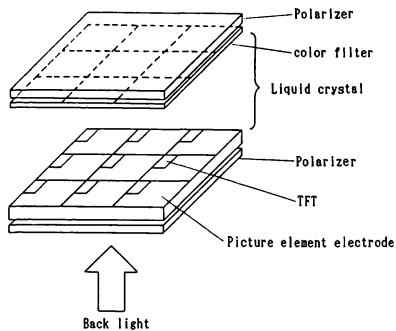


Figure-1: Schematic drawing of LCD panel

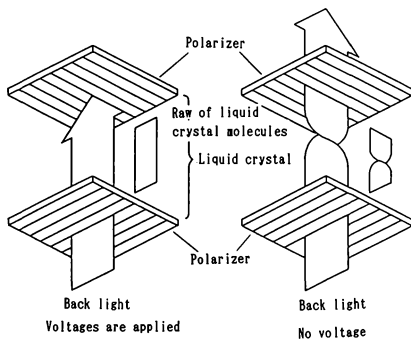


Figure-2: Principle of operation of the LCD

Table-1 Features of LCD display

- | | |
|-----|--|
| (1) | Compact and lightweight |
| (2) | High voltages are unnecessary |
| (3) | Electric power saving |
| (4) | Excellent visibility against ambient light |
| (5) | Neither distortion nor color convergence |
| (6) | Explosion-proof structure is unnecessary |
| (7) | Viewing angle is limited to some extent |
| (8) | The working speed of liquid crystals are affected readily by temperatures. |

Table-2 Characteristics of LCD FPD

	Item	value
1	Display size	5 inch x 5 inch
2	Display color	16 colors
3	Contrast	more than 10
4	Brightness	0.5 ~ 80 ft.L
5	Viewing angle	60° (horizontally)
6	Refresh rate	75 Hz
7	Data update rate	25 Hz
8	Power supply	AC 115V / 400Hz / 70W

3. Constitution of Evaluation Test Unit

An FPD unit for evaluation was installed on an instrument panel in front of the main pilot seat of a flight simulator for the NAL's research and development work. The simulation block diagram is shown in Figure-3.

3.1 Modification of Instrument panel

A conventional ADI (Attitude Director Indicator) was removed from the instrument panel in front of the main pilot seat of the flight simulator, and a D/U (Display Unit) of the FPD unit was installed on the instrument panel instead of the ADI. Figure-4 shows the instrument panel equipped with the D/U. Also, the conventional speed indicator, altimeter, rate of climb indicator, etc., are installed as they stand at their existing positions. An instrument panel in front of the co-pilot seat is not modified to make a comparison between LCD type instruments and conventional type instruments.

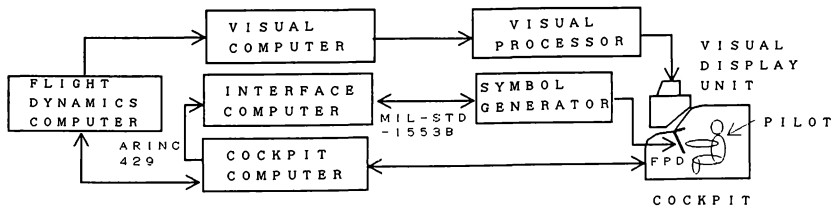


Figure-3: Block diagram of evaluation test system

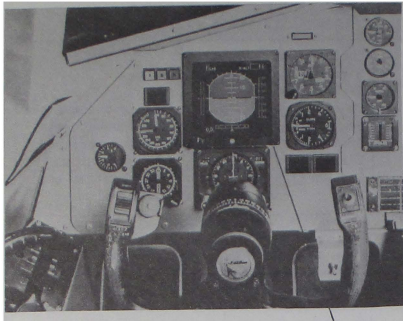


Figure-4: Cockpit arrangement
(Main pilot side)

3.2 FPD Unit

Figures-5 and -6 show a sketch drawing and a block diagram of an FPD unit, respectively. The FPD unit used in this evaluation test consists of (1) interface computer for receiving data transmitted from the simulator system, (2) S/G (Symbol Generator) which plots images on the basis of the data, and (3) D/U which is an indicating section.

(a) Function of Interface Computer

A protocol of data transmitted from the flight simulator unit conforms to the ARINC-429 specification, and the I/F specification of the S/G conforms to the MIL-STD-1553B. The interface computer possesses a function whereby data received with the ARINC-429 format are converted into the MIL-STD-1553B format and are transmitted to the S/G.

(b) Constitution and Function of S/G

A block diagram of the S/G is shown in Figure-7. The S/G consists of I/F section based on the MIL-STD-1553B, computing section using a microprocessor, GC (Graphic Controller) for controlling the image plotting work, and I/F section with D/U. The computing section outputs image plotting commands to the GC on the basis of data received from the I/F section, and the GC outputs control signals of the LCD to the D/U on the basis of the image plotting commands. The S/G used in this evaluation test possesses the following features.

- Composition of screens by window circuit
- Reduction in load of image plotting by adoption of hardware in painting function
- Work of precise plotting respective R, G, and B pixels by independent control

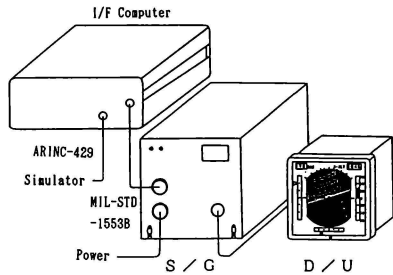


Figure-5: LCD FPD unit

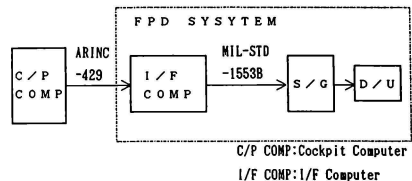


Figure-6: Block diagram of FPD unit

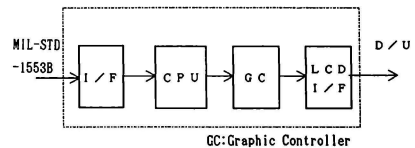


Figure-7: Block diagram of S/G

(c) Constitution and Function of D/U

A block diagram of the D/U is shown in Figure-8. The D/U consists of a buffer for receiving LCD control signals transmitted from the S/G, the LCD/C (LCD Controller) for controlling a transistor of every pixel on the LCD.

Fluorescent lamps are used as a back light because of brightness, color, luminous efficiency, etc. The luminance is controlled with an L/C (Light Controller). It is possible to automatically adjust the luminous intensity by means of signals transmitted from an external light sensor, and to manually adjust that by means of volumes.

(d) Connection with Simulator System

The interface computer is connected to the cockpit computer, and is used to input data for indication. The connection system conforms to the ARINC-429 specification, and the data transmitting rate is 25 cycles per second.

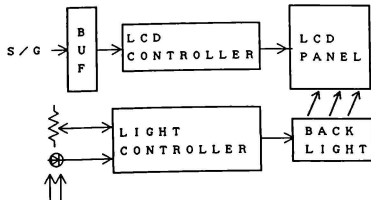


Figure-8: Block diagram of D/U

4. Evaluation Test

4.1 Flight Dynamics Simulation Model

The STOL (Short Take-Off and Landing) experimental aircraft "Asuka" was used as a flight dynamics simulation model.

4.2 FPD Display Format

Figure-9 shows an FPD display format used in this evaluation test. This format is developed for technical research on the Asuka, and is used to indicate attitude, air speed, altitude, rate of climb, distance, localizer, glide slope, etc.

4.3 Evaluation Items

A test on functions and performances of the hardware was conducted, and results obtained from the test were evaluated. Also, the FPD display format was evaluated in respect of the following items about symbols, lines and alphanumerics.

- (1) Size and width
- (2) Tint of color
- (3) Smoothness of movement
- (4) Easiness to read

Table-3 shows an example of an evaluation sheet used in this test.

4.4 Method of Conducting Evaluation Test

The evaluation test is conducted in sequence of the conventional take-off by the CTOL (Conventional Take-off and Landing) procedure, climbing, change of procedure from CTOL to STOL, air work, approach by the STOL procedure, and landing by the STOL procedure. This sequence is repeated some times, and the grades and comments on respective items are entered in the evaluation sheet.

4.5 Evaluation Pilots

The evaluation test was conducted by a total of 23 evaluation pilots (2 research pilots of the NAL, 12 test pilots of aircraft manufacturers, and 9 test pilots of airline service companies).

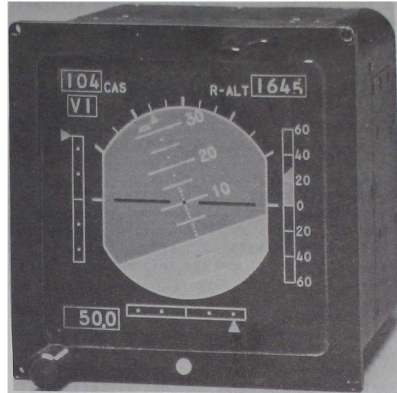


Figure-9: Display format

ITEM	SCALE							COMMENT
	1	2	3	4	5	6	7	
1 Horizontal Line								
2 Airplane Symbol								
3 Pitch Scale								
4 Bank Angle Scale								
5 Bank Angle Pointer								
6 G/S, LOC Scale								
7 G/S, LOC Pointer								
8 Vertical Speed Scale								
9 Vertical Speed Pointer								
10 Air Speed Indicator								
11 Altitude Indicator								
12 G _H , V _s , V _R , V ₂ Indicator								
Remarks	1: Good 4: Unnecessary to modify 7: Necessary to modify - Please put a mark "O" on the scale and write a comment.							

Table-3: Example of sheet for test

5. Results of Test and Their Studies

Tables-4, -5, and -6 show grades of test items (1), (2), and (3). These Tables are plotted in distinction from each other to show the difference between the evaluation of test pilots (mark●) of aircraft manufacturers and that of pilots (mark■) of airline service companies. These marks are common to all the Tables. Also, the number of pilots of the NAL is included in that of pilots of the aircraft manufacturers.

5.1 Function and Performance of Hardware

Table-4 shows a result of evaluating the hardware.

(a) Screen Size

The size of the LCD panel used in this evaluation test is 5 x 5 inches. If this LCD panel is used as an EADI (Electronic Attitude Director Indicator), it is believed that the LCD panel has no problem. However, when the LCD panel indicates other flight information on its screen as an integrated display, the area of the screen is small and the information will be limited.

(b) Resolution of Screen

The screen has no serious problem, but the S/G used in this evaluation test has no antialiasing function when oblique lines are plotted on the screen, so that visibility of details is low. With regard to this point, it is considered to be possible to get better image quality by adding an antialiasing function to the FPD unit.

(c) Brightness of Screen

Brightness of the screen is sufficient at the indication with high luminance, but brightness of the overall screen was nonuniformity, because a back light diffusing function was insufficient. Especially, it was clarified that when luminance was lowered, brightness of the overall screen would be conspicuously nonuniformity and visibility would be deteriorated. Accordingly, it is considered to be necessary to improve the back light mechanisms in the future.

(d) Adjustment of Luminance

It is recognized that there is no practical problem in the automatic and manual adjustments of luminance against ambient light, but flickers are occurred from the screen at around the minimum luminance in the lowering of luminance on the assumption of night flight. Accordingly, it is necessary to realize a function for preventing flickers from being generated at around the minimum luminance and

ITEM	SCORE						
	1	2	3	4	5	6	7
Viewing Area	● ■ ■	● ●	●	● ● ● ● ■ ■ ■ ■ ■ ■ ■ ■	● ●		
Resolution	● ● ■ ■	● ●	■ ■	■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■	■ ■		●
Brightness	● ● ● ● ■ ■	● ●	●	■ ■ ■ ■ ■ ■ ■ ■ ■ ■		■ ■	● ●
Brightness Control	● ● ● ■ ■ ■ ■	●		■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■	●		● ● ● ●
Visibility Against Ambient Light	● ● ● ■ ■ ■ ■ ■ ■ ■ ■	●	■ ■ ■ ■	● ● ● ● ■ ■ ■ ■ ■ ■ ■ ■			
Number of Colors	●	●	● ●	● ● ● ● ■ ■ ■ ■ ■ ■ ■ ■	●	●	■ ■ ■ ■ ■ ■ ■ ■
Pixel Defect	● ● ●		■ ■	● ● ● ● ■ ■ ■ ■ ■ ■ ■ ■	● ●	■ ■ ■ ■ ■ ■ ■ ■	

Viewing Angle (Horizontally)	Ideal Angle		Maximum Angle
	Airplane Maker		
	5 5 . 7 °		6 5 . 0 °
Air Line		5 0 . 0 °	6 2 . 8 °

● ■ : 5 Pilots
 ● ■ : 1 Pilot
 ● : Airplane Maker
 ■ : Air Line

Table-4: Result of test about hardware performance

another function for continuously adjusting luminance. It was also clarified that when high luminance is changed to low luminance, there is large change of color tone on CRT display, but it is a little on the FPD unit. So, it is considered that there is no practical problem in the color tone at low luminance.

(e) Visibility

The FPD unit is more excellent than CRT display unit in visibility. Especially, when intensive light like the sunlight is irradiated to screens of CRT display units, it is difficult to identify the contents indicated on these screens, because of reflected light. It was confirmed that characteristics of the FPD unit was more excellent than those of the CRT display unit, because the visibility of the FPD unit is sufficiently high.

(f) Number of Colors

It is considered to be possible to sufficiently use the FPD unit as an indicator, because there is no practical problem in eight basic colors, red, blue, green, yellow, cyan, magenta, white, and black. It will be necessary to display full color to indicate video images. So, it will be necessary to establish a technology for generating neutral tints and to modify the hardware including the LCD panel.

(g) Pixel (Picture Element) Defects

The yield at manufacturing of LCD panels depends on the number of picture element defects. It is very difficult to manufacture LCD panels with no picture element defect by using present LCD manufacturing technologies.

Line defects and area defects of LCD panels have not yet been defined, but such LCD panels were selected on the basis of the following items: (1) the LCD panel shall not have any line defects and plane defects, and (2) the number of picture element defects shall be within 20 at a total of 600,000 picture elements.

As a result, it was clarified that there was no serious problem in LCD panels unless picture element defects occurred at a portion of these LCD panels. It is considered to be necessary to uniformly define picture element defects, because it is anticipated that LCD panels will be used widely in various fields in the future.

(h) Lateral Angle of Visibility

The deterioration of visibility from obliquely lateral directions can be cited as one of the demerits of LCD panels, however, the visibility from an usual operating range of a pilot is sufficient. Figure-10 shows an angle in the case when the EADI in front of the left pilot seat is seen from the right pilot seat in a large jet plane. The angle is about 60° . The maximum allowable angle of visibility in the horizontal direction was obtained from this evaluation test. This angle is 58.4° on an average, and is considered to be almost good from the standpoint of present performances. However, it is considered to be necessary to ensure wide angle of visibility.

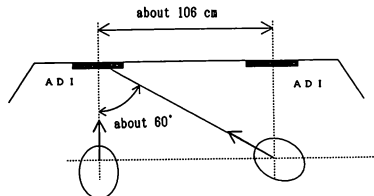


Figure-10: Viewing angle from co-pilot seat to the ADI on the instrument panel front of main pilot seat (case of large transport plane)

5.2 Display Format

(a) Size and Width of Symbols, Lines, and Numerical Characters

Table-5 shows results of the evaluation. Basic display formats were almost good, but particularly, the airplane symbol, pitch scale, rate of climb scale, and rate of climb pointer were low scored on the whole. Therefore, it can be appreciated that these should be reconsidered.

(b) Tint of Color of Symbols, Lines, and Numerical Characters

Table-6 shows results of the evaluation. It was clarified that the ADI background lower sphere as ground (neutral tint: brown), airplane symbol, glide slope, localizer, and rate of climb pointer were low scored.

Amber was used as a color of the ADI background lower sphere to show the ground, but the indication of neutral tints was insufficient. In addition, patterns striped vertically on sphere gave a sense of incompatibility to pilots. It is considered to be necessary to modify the functions for indicating neutral tints.

Unevenness of formats was pointed out in respect of the glide slope, localizer, and rate of climb pointer.

Indication formats for this evaluation test were prepared individually for technical research on Asuka. For this reason, it was difficult to understand the contents of the indication. It is considered to be desirable that indication formats for civilian aircraft should conform to SAE ARP-11, and should be unified internationally in the future.

(c) Smoothness of Movement of Symbols, Lines, and Numerical Characters

The data transmitting period was 25 cycles per second in this evaluation test, because of limitation of performances of the simulator computer, but it is considered that the refresh period of the indication screen must be about 30 cycles per second.

The speed, altitude, and distance were digitally indicated by means of a rotating drum system so that pilots do not feel any senses of difference between conventional indications and new ones. Particularly, the altitude was indicated at a unit of 1ft so that pilots can read even values of low altitudes. It was clarified that the rotating speed of the drum was increased and reversely the visibility was decreased in accordance with increase in rate of climb.

Comments on the pitch scale are as follows: (1) the line is very thin, and (2) the jaggy portions generate conspicuously in the case when the airframe takes small banks. So that it would be better to change a dot every 1° to a short lateral line.

ITEM	SCORE						
	1	2	3	4	5	6	7
Horizontal Line	●	●●	●●	●●●	●●	●	
Airplane Symbol	●	●●	●●	●●●	●●●	●●	■
Pitch Scale	●	●●	●●	●●●	●●	●	■
Bank Angle Scale	●●	●	●●	●●●	●●	●	●
Bank Angle Pointer	●	●●	●	●●●	●	●	
G/S, LOC Scale	●●●	●	●●	●●●	●	●	
G/S, LOC Pointer	●●●	●	●●	●●●	●	●	■
Vertical Speed Scale	●●●●	●	●●●	●●●	●●	●●●	
Vertical Speed Pointer	●●●●	●	●●	●●●	●●	●	■
Air Speed Indicator	●●●●	●●	●	●●●	●	■	■
Altitude Indicator	●●●●	●●	●	●●●	■	■	■
GA, V ₁ , V _R , V ₂ Indicator	●●	●	●●	●●●	■	■	■

● ■ : 5 Pilots ● : Airplane Maker
 ● ■ : 1 Pilot ■ : Air Line

Table-5: Result of test about the symbol shape

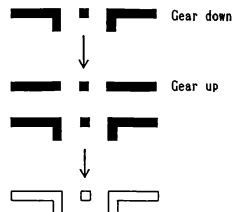
ITEM	SCORE						
	1	2	3	4	5	6	7
Sphere (Up Side)	●●●●	●●	●●	●●●	■		
Sphere (Down Side)	●	●●	●●	●●●	■	■	●●
Horizontal Line	●●●	●●	●●	●●●	■	■	■
Airplane Symbol	●●	●●	●●	●●●	■	●●	■
Pitch Scale	●●●	●●	■	●●●	■	■	●
Bank Angle Scale	●●●	●●	●●	●●●	■	■	●
Bank Angle Pointer	●●	●●●	■	●●●	■	■	●
G/S, LOC Scale	●●●	●●	■	●●●	■	■	■
G/S, LOC Pointer	●●●●	●●	●●	●●●	■	■	■
Vertical Speed Scale	●●●●	●	■	●●●	■	■	●
Vertical Speed Pointer	●●●●	●	●●	●●●	■	■	■
Air Speed Indicator	●●●●	●	■	●●●	■	■	■
Altitude Indicator	●●●●	●●	■	●●●	■	■	■
GA, V ₁ , V _R , V ₂ Indicator	●●	●	●●	●●●	■	■	■

● ■ : 5 Pilots ● : Airplane Maker
 ● ■ : 1 Pilot ■ : Air Line

Table-6: Result of test about the symbol color

The airplane symbol was based on a shape shown in the following drawing. In order to clarify the state of landing gears in and out intuitive format was adopted. However, this function was eliminated, because it is difficult to judge the longitudinal direction at the inverted flight.

In addition, when the airplane symbol is overlapped with the pitch scale, the pitch scale will be behind the airframe symbol and cannot be seen clearly. Accordingly, as shown in the following drawing, the format of airplane symbol was changed from painted black to airplane frame was white and inner frame was transparenence.



Next, with regard to the indication of rates of climb, at first the full scale was decided to be $\pm 6,000$ feet per minute, but for convenience sake, the full scale was set so that it automatically changes to $\pm 3,000$ feet per minute at an altitude of 4,500 feet or more and to $\pm 1,500$ feet per minute at that of 4,500 feet or less, because the full scale of $\pm 6,000$ feet per minute is a very large value and it is difficult to precisely control the rate of climb at the take-off and landing. However, the following items were clarified : (1) it is difficult to precisely read indicated values, and (2) it is difficult to operate the plane while retaining altitudes in the vicinity of a rate of climb of zero, because when the full scale is $\pm 1,500$ feet per minute, the indication gain is very high.

(d) Readability of Symbols, Lines, and Numerical Characters

It is considered that readability is evaluated as an overall result of the shape, size, tint of color, smoothness of movements, etc. As previously mentioned, as a result of evaluating the airframe symbol, pitch scale, rate of climb scale, pointer, indication of altitude, etc., these were clarified to be illegible. It is considered that indications must be modified so that pilots can readily read the above items by restudying the digital indication system and tints of colors including neutral tints and by overall reviewing the indication format on the basis of the above result.

6. Future Subjects

6.1 Hardware

It is considered to be necessary to establish technologies for indicating full colors including neutral tints, those for erasing jaggy portions, those for wide angles of visibility on large screens, and functions for finely adjusting low luminance, as future technical subjects.

6.2 Indication Format

It is anticipated that many actual planes will be equipped with electronic integrated instruments employing the LCD, etc., in the future, and at that time, it is desirable that the shape, tint of color, etc., of each symbol should conform to the SAE ARP, etc., to unify the indication formats.

7. Conclusion

As a result of conducting the evaluation test by the flight simulator, it was confirmed that the LCD could be used sufficiently as an indicator for aircraft.

In the future, we will carry out research on further increase in performances and miniaturization of the LCD and will develop new LCD's installed in actual planes while attaching importance to rise in environmental resistance and reliability on the basis of the above results obtained from the evaluation test.

Reference

- 1)Matsumoto ; Liquid Crystal Electronics, pp35-pp36, OHMU-SHA, 1986
- 2)H.Kawahara et al. ; Liquid Crystal Flat Display Evaluation Test by Flight Simulator, Proceedings of The 27th Aircraft Symposium, pp510-pp513, in JAPAN, 1989. 10
- 3)T.Udagawa et al. ; Development of Flat Panel Display for Aircraft, Proc. of The 27th Aircraft Symposium, pp506-pp509, in JAPAN, 1989. 10
- 4)SAE ; Aerospace Recommended Practice, ARP-4102/7, 1987. 11

D-Size Liquid Crystal Flat Panel Display

Evaluation by Flight Simulator

Hiroyasu KAWAHARA*, Kaoru WAKAIRO*, Akira WATANABE*
National Aerospace Laboratory, Japan

Tetsuo SHIMIZU**, Kenji MATSUMURA**, Shinichi OHYAMA**
Japan Aviation Electronic Industry

Abstract

In recent years, the main cockpit of commercial aircraft has been the so-called "glass cockpit", which uses electronic flight instruments with large-size color CRTs. In the near future, cockpit displays using the liquid crystal flat panel display, will be put into practical use and will supersede the CRT display.

Flat panel displays include the LCD, LED(*) display, and plasma display, which have respective advantages and disadvantages.

The liquid crystal flat panel display provides several advantages compared to other displays.

Results of evaluation tests indicate that the liquid crystal display panel has functions and performance necessary for commercial aircraft. This report describes the features and technical issues of the liquid crystal panel display.

1. Introduction

Outstanding among liquid crystal display panels for flight instruments is the TFT(:2) and TN(*3) active matrix type, which has excellent contrast, response, color and viewing angle characteristics.

The color LCD has been upgraded from a multi-color display (8 colors) operated by a TFT binary drive to a full-color display.

- *1 LED: light emitting diode
- *2 TFT: thin film transistor
- *3 TN: twisted nematic
- *4 DPI: dot per inch

2. Technical issues of liquid crystal flat panel display and their solution

(1) Bright and dark environment

The prime issue is to ensure display quality under a high illuminance environment assuming a flight during sunlight and under low illuminance during a night flight.

As compared with the CRT, the liquid crystal display panel features fine readability under direct sunlight and few changes in color tone even at lower luminance display.

A wide range brightness (0.1 FL or less to 10,000FL or more) control function is required to guarantee the good readability in the day and night. of forward field of view in the daytime, and readability of display in the starlight. These requirements depend on the back light performance of the LCD panel.

For this reason, the following developments have been achieved.

(a) High luminance, long life fluorescent lamp (more than 20,000 hours).

(b) Back light driver capable of a wide range of brightness control.

(c) High efficiency, redundancy of back light to structure.

play with a grayscale feature operated by a TFT linear drive.

As for display size, resolution is improved. The LCD panel has been developed to the point where a resolution of 150 DPI(*4) is now available for an ARINC D size display.

Performance has also been improved to the point that it equals that of conventional panels using a color CRT.

At present the small-size liquid crystal flat panel display in 3ATI size has been put into practical use, serving as the display for the Traffic Collision Avoidance System(TCAS).

* Flight Simulation Lab.,
Control and Systems Division
** Electronic Systems Engineering Dep.,
Aerospace Division

Copyright © American Institute of
Aeronautics and Astronautics, Inc.,
1990. All rights reserved.

(2) Readability in the lateral direction

The LCD panel viewed from the side seat (cross cockpit) differs in readability from that viewed from the front. This difference is one of the major disadvantages of the LCD panel.

To solve this problem, a cell gap and color filter structure have been optimized, leading to a horizontal viewing angle of ± 60 degree or more.

(3) Legibility of display

The legibility of symbols, characters, lines, etc. depends on resolution, color mixture, contrast, etc.

(a) Resolution

Human engineering evaluation tests have proven that a resolution of 150 to 170 DPI produces legible characters and lines, and that a higher resolution does not enhance legibility²⁾.

(b) Color mixture

Fig. 1a and 1b show the luminance distribution of the pixel pattern of a high resolution CRT. Fig. 1 proves that the CRT has a structure which is likely to cause color mixture in the human eye.

On the other hand, the LCD panel has square dots arranged in a matrix form, as shown in Figs. 2a and 2b. The luminance is distributed digitally.

For this reason, the LCD panel has the following problems as compared with the CRT.

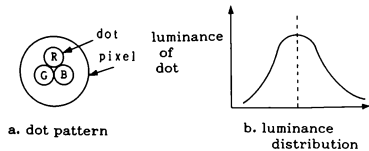


Fig. 1 COLOR CRT

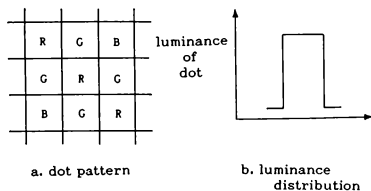


Fig. 2 FLAT PANEL DISPLAY

* Luminance at the boundary between dots is a discriminating feature.

* Color mixture performance degrades depending on the angle of a slant line, causing a color band (particular colors become prominent.)

* In a line drawing, slant lines are jagged.

To cope with the above problems, the following technical developments have been achieved.

* The RGB delta array pattern shown in Figs. 3 and 4 is employed.

* Grayscale display has been attained by combining the LCD driver with an LCD panel capable of setting the grayscale for each dot of the RGB, thus improving color mixture performance and realizing jag-free display (anti-aliasing ^{*5)}).

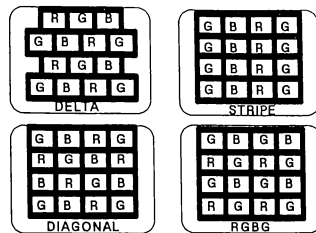


Fig. 3 Various Pixel Arrangements^{*1)}

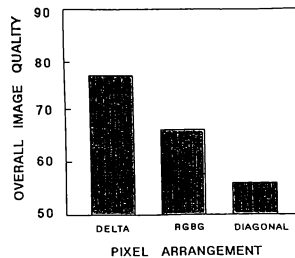


Fig. 4 Pixel Arrangements VS Image Quality^{*1)}

^{*5} Anti-aliasing means that jagged lines are turned into smooth lines, by antialiasing in the human eye, through interpolation of medium grayscale dots by setting a grayscale level for each dot of the RGB. Refer to Fig. 5.

(C) Contrast

An increase in contrast depends on how the black of the background is displayed. This means that leakage of the back light must be reduced to a minimum.

Technical developments have been made to increase contrast through optimization of the LCD panel structure and cell gap, for filming improvements.

- * Contrast in off-axis viewing angle (i.e., cross-cockpit viewing angle).
- * Contrast in extremely low luminance

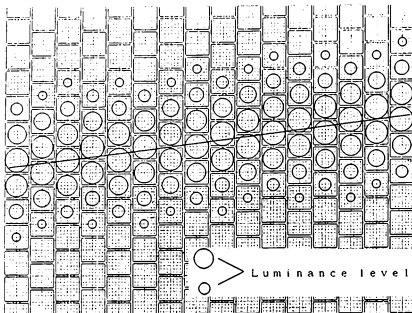


Fig. 5 Anti-aliasing Method

*6 EFIS): Electronic Flight Instrument System

3. System Configuration for Evaluation.

Fig. 6 shows the system configuration for evaluation. The following is a description of each unit of the system.

(1) Liquid crystal flat panel display for evaluation

Table 1 outlines the performance of the liquid crystal flat panel display evaluated.

(a) Primary flight display (PFD)

Figs. 7a and 7b show the format of the PFD which has been prepared, with reference made to the recommended format of SAE-ARP³ and the format based on the EFIS(*6) of MD-11.

(b) Digital map display

Fig. 8 shows an example of the displayed digital map, which covers the following information.

- * 2D terrain image
- * Contour line
- * Terrain data (rivers, roads, railways, etc.)
- * Airport
- * Navigation ground support facilities.

Table 1 LCD Flat panel performance

No.	Item	Performance
1	Size	8" x 8" x 6.5"
2	Effective display size	6.85" x 6.85"
3	Resolution	1024 x 1024 dots
4	Brightness	0.1ft-L - 100ft-L
5	Viewing angle	+/-60 degree (horizontal) 0 - 45 degree (vertical)
6	contrast	> 6
7	Display color	Full colors
8	Jag	Jag-free line

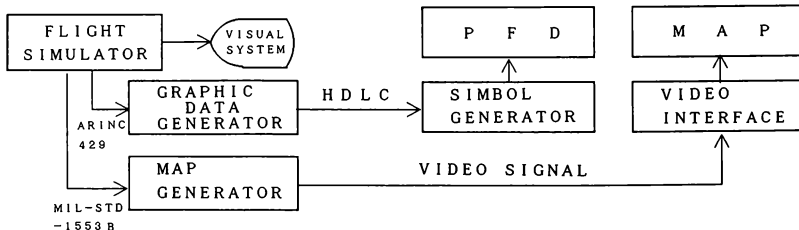


FIG. 6 SYSTEM BLOCK DIAGRAM

(2) Graphic data generator

The graphic data generator receives data necessary for the PFD from the flight simulator computer through the data bus designed to the ARINC 429 specification, and generates the real time graphic command for the PFD.

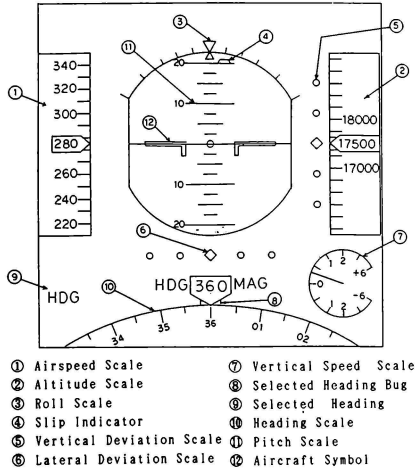


Fig. 7a PFD Format

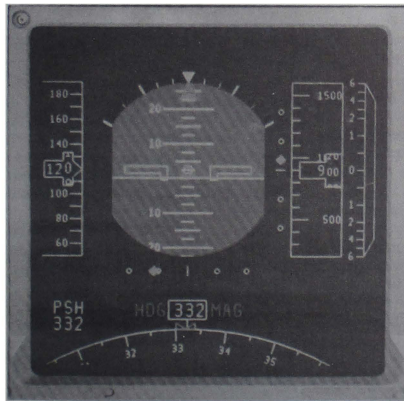


Fig. 7b Photograph of PFD

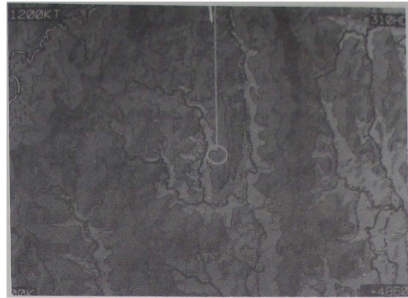


Fig. 8 Photograph of Digital Map Display

(3) Symbol generator

The symbol generator generates image data for the LCD upon receipt of graphic commands from the graphic generator, and transmits the PFD image data to the LCD panel.

(4) Map generator

The map generator stores map data in its data area, and generates the digital map image of the present position in the form of a video signal upon receipt of the aircraft position data from the simulator computer through the data bus designed to MIL-STD-1553B.

(5) Video interface

The video interface converts the video image from the map generator into image data before transmitting them to the LCD panel.

4. Evaluation by Flight Simulator

4.1 Purpose of evaluation

During bench evaluation the liquid crystal flat panel display thus developed has demonstrated much the same function and performance as the conventional CRT type display.

To pursue the evaluation farther, pilots were asked to evaluate hardware and display quality (format), in a flight simulator.

4.2 Cockpit instrument panel for simulation

Two D-size displays were installed for simulation. One was the LCD panel installed on the simulation cockpit instrument panel as the primary flight display (PFD), and the other was the map display (digital map) installed on the center instrument panel.

Fig. 9 shows the whole view of the cockpit and the LCD panel installed.

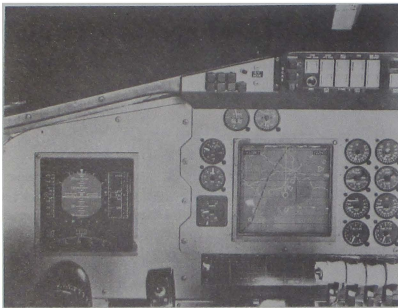


Fig. 9 LCD Panel Installation in Cockpit

4.3 Flight simulation motion model

In the flight simulation, the model of the NAL experimental STOL "ASKA" was used as the flight motion model.

For the purpose of simulation take-off and landing, the conventional take-off and landing configuration was used.

4.4 Evaluation method and items

In evaluation, the usual take-off, climb, air work, approach, and landing were repeated several times, and pilots were requested to mark the evaluation sheet given in advance for each evaluation item and enter their comments on said sheet.

Table 2 shows the evaluation items.

The pilots marked the evaluation sheet according to the scoring system as defined below.

Table 2 Pilot Evaluation Item

1	Function and performance of hardware
2	Character, thickness and size of line
3	Color matching of symbol, character, etc.
4	Smooth screen movement
5	Legibility of symbol and character

4.5 Evaluation pilots

Eighteen pilots were assigned to the evaluation test, including pilots from NAL, aircraft manufacturers, and airline companies.

5. Evaluation and consideration

Scores (marks) for each item and comments were given by the pilots, as listed below.

Evaluation sheet

score	1	2	3	4	5
evaluation	to be improved		usable		excellent

(1) Function and performance of hardware

Fig. 10 shows scores given to the function and performance of hardware. The main comments of the pilots are as shown below.

(a) PFD

- * The LCD panel has display quality equivalent to that of the conventional CRT panel.

- * Color matching remains unchanged even in full dimming status, but contrast should be improved.

- * The LCD panel has excellent readability, providing a sufficiently wide viewing angle (more than ± 60 degree) when viewed from the co-pilot's seat.

- * The LCD panel has particularly excellent sunlight readability as compared with the CRT panel.

(b) Digital Map

- * The zoom function is effective.
- * Roads, railways, etc. should preferably be limited only to trunk lines on display.
- * The distance marker enables the pilot to predict the time of arrival during a flight, and should therefore be provided by all means.
- * The pilot should be allowed to select content of Map depending on intended use.

(2) Display quality

(a) Symbol, character, thickness and size of line

Fig. 11 shows scores given to the symbol, character, thickness and size of the line. Main comments were given as listed below.

- * Jaggy lines are less noticeable.
- * Readability of thin lines requires improvement.
- * Vertical speed display is too small.
- * The aircraft symbol requires modification in both width and length.

(b) Color matching of symbol, character, etc.

Fig. 12 shows scores given to color matching of the symbol, character, etc. The main comments of the pilots are as listed below.

- * Color matching is acceptable, on the whole. However, small symbols should be given more contrast to the background color.
- * The pointer should be made to stand out more.

(c) Smooth screen movement

Fig. 13 shows scores given to smooth screen movement.

The main comment of the pilots is that symbols and characters move smoothly.

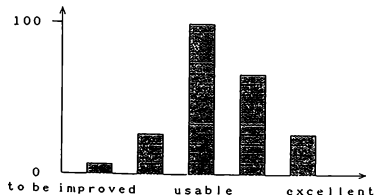


Fig. 10 Function and Performance of hardware

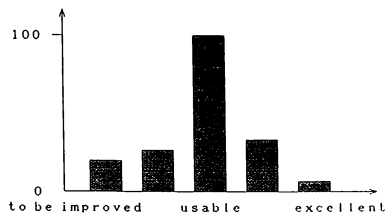


Fig. 11 Symbol, character and size of line

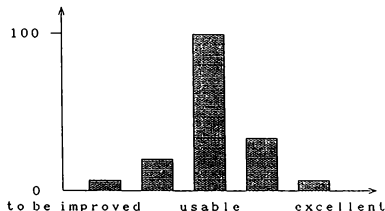


Fig. 12 Color matching of symbol, character

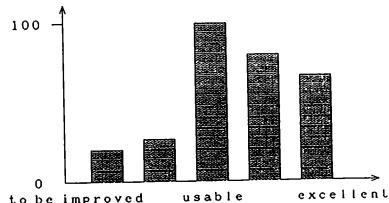


Fig. 13 Smooth movement

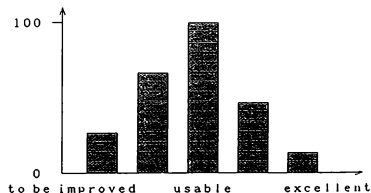


Fig. 14 Readability of symbol and character

(d) Legibility of symbol and character
Fig. 14 shows scores given to legibility of the symbols and characters.

The main comments of the pilots are as shown below.

* The symbols and characters are legible on the whole, but they could be made more legible through sophisticated design of the format, color coordination, etc.

6. Conclusion

The main result obtained by the evaluation test indicates that the D-size liquid crystal flat panel display is acceptable as a civil aircraft display.

It needs, nevertheless, the following improvements to attain more practical use.

- (1) clear black color realization,
- (2) high resolution (more than 170 DPI),
- (3) efficient processing capability,
- (4) a large screen display in thin package with small bezel,
- (5) more readable display formats

As a result of these improvements, in near future, the LCD flat panel display will supersede the CRT as a cockpit main display.

References:

- 1) Rupp, J. A. : "Flat panel displays for Avionics", Technical paper of display materials seminar, Oct. 1988.
- 2) Krantz, J. H. and Silverstein, L. D. : "Color matrix display image quality, the effect of luminance and spatial sampling" Technical paper of SID, 1990.
- 3) SAE-ARP : ARP 4012/7 Society of Automotive Engineers Inc. Aerospace Recommended Practice, July 1988.

DESIGN, DEVELOPMENT AND TESTING OF AN AMBIENT LIGHTING SIMULATOR FOR EXTERNAL ILLUMINATION OF A TRANSPORT SIMULATOR COCKPIT

Vernon M. Batson and Lawrence E. Gupton
Crew/Vehicle Interface Research Branch
NASA-Langley Research Center
Hampton, Virginia 23665

ABSTRACT

Researchers at NASA-Langley Research Center several years ago began to examine concepts for a facility which could simulate the full range of lighting conditions encountered in flight. The purpose of this facility was to evaluate advanced technology display devices in a cockpit environment. The Aircraft Cockpit Ambient Lighting Simulation System has been developed to meet that need. The ACALSS surrounds a wide-body part-task simulator cockpit, the Advanced Display Evaluation Cockpit, and interfaces with a VAX 11/780 computer, which is used to host both a math model of a modern transport aircraft and a solar motion model (which animates a servoed sun simulator).

In defining the requirements for this facility, the ambient lighting conditions which were considered to be worst-case, in terms of a pilot's ability to use these advanced technology devices, were defined. Several concepts were evaluated and an efficient design using an ellipsoid reflector and HMI Fresnel luminaires was selected for implementation. Illumination levels at the pilot's instrument panel can be controlled from darkness to well over 10,000 footcandles and luminance in the forward field-of-view can be controlled to as high as 18,000 footlamberts over a small area. A study which was recently completed in this simulator used the over-the-shoulder sun simulation capability in evaluating pilot/vehicle performance as a function of lighting levels for each of three different display devices.

INTRODUCTION

Before the introduction of the cathode ray tube (CRT) into the modern transport aircraft cockpit during the 1980's,^{1,2,3,4} cockpit instrumentation had consisted mainly of single-purpose, dedicated electromechanical instruments and controls. These reflective electromechanical displays could easily be seen by the pilot in direct sunlight and, through back or edge lighting, in subdued light. Consequently, these devices could be installed in the cockpit with little concern for their visibility under high ambient illumination. The CRT, however, being an emissive device, suffered some degradation under direct sunlight.^{5,6} Additionally, some of the developmental flat-panel display devices which were anticipated to become competitive with the CRT for use in the cockpit were emissive devices and hence also degraded in direct sunlight.^{7,8} Clearly, a method of evaluating these display devices prior to installing them into the cockpit was needed.

Recognizing this need for a facility which would enable researchers to evaluate candidate display technologies in a simulator cockpit prior to aircraft installation, researchers at NASA-Langley Research Center (LaRC) began in the mid-1970's time-frame to develop concepts for a lighting facility which could simulate the full range of lighting conditions encountered in flight. The Aircraft Cockpit Ambient Lighting Simulation System (ACALSS) is the result of that effort. The ACALSS surrounds a wide-body, part-task simulator cockpit, the Advanced Display Evaluation Cockpit (ADEC) (Figure 1), and interfaces with a

VAX 11/780 computer, which is used to host both a math model of a modern transport aircraft and a solar motion model (which animates a servoed sun simulator).

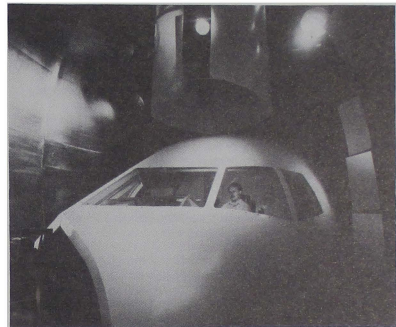


Figure 1. The Aircraft Cockpit Ambient Lighting Simulation System (ACALSS) Surrounding the Advanced Display Evaluation Cockpit (ADEC)

DESIGN AND DEVELOPMENT

The ambient lighting conditions which were considered to be the worst-case in terms of the pilot's ability to readily interpret the information being presented to him on the display were:⁹

1. Direct sunlight shining onto the display surface through a side window or over the pilot's shoulder,
2. Sunlight reflecting from a white shirt onto the display surface,
3. The sun in the pilot's forward-field-of-view (FFOV) at a low elevation angle above a deck of clouds, and
4. Darkness.

The ability to simulate these four conditions in the ambient lighting simulator was established as a minimum requirement for the simulator. As the design parameters necessary to meet these requirements were being developed, it was decided to incorporate a lightning flash simulator to add to the ACALSS the capability of conducting pilot/vehicle studies with the dark-adapted eye under simulated thunderstorm conditions.

Copyright © 1990 by the American Institute of Aeronautics and Astronautics, Inc. No copyright is asserted in the United States under Title 17, U.S. Code. The U.S. Government has a royalty-free license to exercise all rights under copyright claimed herein for Governmental purposes. All other rights are reserved by the copyright owner.

The minimum luminance and illumination requirements needed to meet the ambient lighting conditions outlined above were:^{1,6,10}

1. Sunlight illuminating the side window of the cockpit at 10,000 footcandles, shining onto the instrument panel,
2. A diffuse cloudy sky at a luminance of 10,000 footlamberts,
3. A 2,000 footlambert sky with the sun (at a sufficient luminance level to cause the pilot's pupil to stop down) in the FFOV,
4. Darkness, and
5. Lightning flashes (at a level sufficient to cause a pilot to lose his dark-adapted vision) during darkness.

The design goals for this facility incorporated these discrete requirements, as well as the capability to simulate the full range of ambient lighting conditions encountered in flight. Realism was to be sought for, in terms of a continuous FFOV and gradual changes in lighting conditions. Efficiency of design and operation were also essential.

EVALUATION OF CONCEPTS

A study funded by the Air Force to evaluate the feasibility of a facility which would have the capability to simulate all lighting conditions experienced during military flights¹¹ provided useful input to early design concepts for the ACALSS. Simulation of the sun and sky in this study were designed to be separate elements. The sky simulation design was what might be roughly called an "integrating sphere" concept, with two designs evaluated -- one an externally illuminated translucent dome and the other an internally illuminated opaque reflective dome with a diffusing surface. Because of the large physical size of the domes in this study (approximately 40 feet in diameter), and the resulting large number of luminaires required to illuminate them, the cost of construction and operation of a similar facility for LaRC's researchers would have been prohibitive. Consequently, only scaled-down versions with smaller domes were considered.

Figure 2 shows one preliminary concept for the ACALSS. Sky simulation consisted of a dome-shaped grid supporting an array of luminaires, with each individual luminaire covered with a light-diffusing material. Several commercial solar simulators were to be mounted on a vertical curved bar which would move along a circular track in the floor inside the dome to give simulated solar elevation and azimuth angles. (These solar simulators used xenon lamps which required rigid vertical mounting. Therefore, multiple simulators were required to cover the windscreen area in segments.) One of the problems with this concept was the "used car lot" sky appearance caused by multiple image sources mounted close to the diffusers.

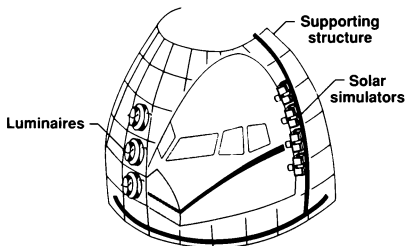


Figure 2. Preliminary Concept 1

Another concept (Figure 3), representing a much more economical approach, used several moveable arrays of luminaires and reflective panels which would be positioned for each experiment as desired and would remain static during an experimental run. The solar simulator could be rolled into position and aimed as desired. However, this simulator lacked realism, as did the previous concept, and did not lend itself to a dynamic simulation.

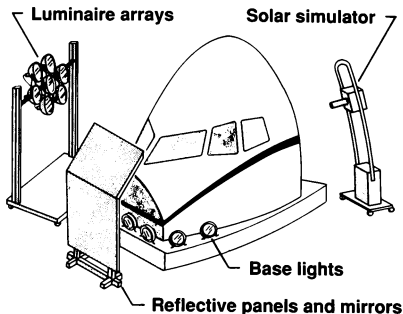


Figure 3. Preliminary Concept 2

A third concept, although similar in principle to Figure 3, refined some of the individual components. The simulated sky panel used lamps mounted in linear parabolic reflectors to give a more uniform light distribution across the panel face. The solar simulator used a custom-designed circular parabolic reflector which could be servo-positioned along a vertical circular track to simulate varying sun elevation angle. Again, only portions of the simulated sky could be illuminated at one time.

In an effort to simulate continuous sky and to provide an automated solar simulator, a fourth concept was developed. In this concept, a dome of diffusing plastic surrounded the cockpit. The luminaires, similar in design to those in the third concept, were mounted in movable panels behind the dome and could be repositioned and diffused to provide various levels of sky brightness. The servo-driven sun would move along a curved vertical track with the track moving horizontally along the floor, pivoted above the center of the cockpit. Several refinements of these concepts were also examined before the design process took a different tack.

DEVELOPMENT OF FINAL DESIGN

In each of the foregoing concepts, realism exists at the expense of efficiency or efficiency exists at the expense of realism. Since efficiency of both design and operation were driving factors of the design, a closer look was taken at the various geometries involved. It was expected that a pilot's performance with a particular display device would be a function of the amount of light illuminating the display and the luminance of the "sky" in his FFOV. Therefore, the windscreen and instrument panel became the primary areas of interest for an efficient design. It became apparent that by focusing light at these two areas, a more efficient design (i.e., one that didn't try to illuminate the entire volume within a dome to the same illumination level) could be realized.

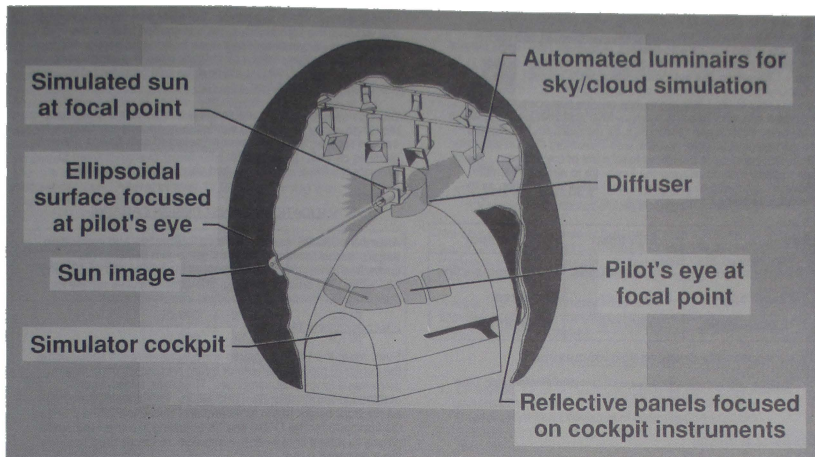


Figure 4. ACALSS Final Design

Ellipsoidal Reflector

Since an ellipse has the characteristic that a beam of light originating at one focal point and reflecting off the side passes through the other focal point, ellipsoidal surfaces were incorporated into the design. For the FFOV, it was desired to present a continuous surface to the pilot in order to realistically simulate sky and cloud cover. Therefore, an ellipsoid of revolution (i.e., an ellipse in the vertical cross section and a circle in the horizontal cross section), with the major axis vertical, was determined to be the most effective geometry for the FFOV (Figure 4).

Two points were defined within the cockpit: the pilot's eye reference point (PERP) and the pilot's instrument reference point (PIRP). The physical locations of the PERP and the instrument panel had been established by the designer of the ADEC, based upon human engineering considerations for an "average" pilot. The PIRP was now defined as a point on the pilot's instrument panel directly in front of him at the perpendicular intersection of the panel and his downward line of sight. For the FFOV, the PERP was chosen to be located at the lower focal point of the ellipsoid. The upper focal point was chosen as the location of the "sun" light source and was designed to be 6 feet above the PERP. The horizontal diameter of the ellipsoid (minor axis) was chosen to be 22 feet. The same light source was to be used to simulate the sun in the FFOV as well as the over-the-shoulder shafting sun. Therefore, for the PIRP, a second ellipsoid with one focal point at the light source and the other focal point at the PIRP was needed. Due to limited space available in the simulator room, as well as less emphasis on a continuous reflective surface for the over-the-shoulder illumination, adjustable segments of ellipsoidally-shaped reflective panels were used. These panels were swivel-mounted behind the pilot's shoulder to provide greater flexibility in aiming the light into the cockpit instrument area. Both the ellipsoid and the panels were constructed of molded fiberglass and then covered with a reflective material.

Sun Simulator

During the 1970's, commercially-available sun simulators normally used a xenon lamp as a source. Typically, these lamps had a restricted stationary mounting orientation and a sizeable output in the infrared. Moreover, the sun simulator output was only a few inches in diameter. Since the design goals called for the sun in the FFOV as well as over-the-shoulder, and the spot size for illuminating the instrument panel had to be large enough to cover one or more instruments, xenon-based sun simulators were ruled out. Another lamp, the HMI, which is still commonly used in luminaires for the movie and TV industry, has a spectral distribution close to that of daylight¹² (Figure 5) and a low infrared output as well. Fresnel luminaires which use the HMI lamp were commercially available and were relatively efficient at approximately 100 lumens per watt. Two such luminaires, a 2500 and a 4000 watt unit, had been obtained to aid in conducting a study in the ADEC prior to design of the ACALSS.¹³ This made it possible to evaluate these luminaires in-house.

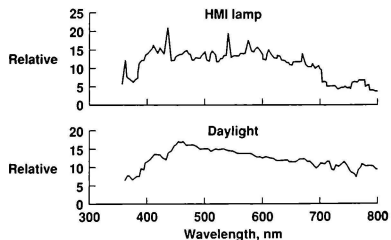


Figure 5. Relative Spectral Distribution of (a) HMI Lamp and (b) Daylight at 6500° K (Source: OSRAM)

Since it was desired to simulate the range of solar intensities experienced from sunrise to sunset, a method of varying the intensity of the source was needed. However, HMI lamps cannot be dimmed conventionally by simply varying voltage or duty cycle. Therefore, the luminaires which were chosen have a focus control which varies the spot size and intensity of the illuminated area (Table 1). In order to simulate relative sun position as the part-task simulator is being flown, a 4000 watt HMI luminaire which was automated in pan (horizontal), tilt (vertical), and focus, and had an automated color scroller, was chosen as the sun simulator and was mounted at the upper focal point of the ellipsoid. These automated features allowed the luminaire to simulate azimuth, elevation, and time-of-day as well as relative sun position as the aircraft was being flown.

	Full flood	Full spot
Maximum output	74 footcandles	1391 footcandles
1/2 peak angle*	64 degrees	8 degrees
1/10 peak angle	76 degrees	16 degrees

*Where intensity drops to 50% of maximum

Table 1. 2500 Watt HMI Fresnel Luminaire Photometric Data
(Measured at 31.62 feet)

Sky/Cloud Simulation

Simulation of the luminance of the sky and cloud cover were considered to be of primary importance, with simulation of the their color content desirable, but of secondary importance. To illuminate the segment of the sky which was viewable by the pilot through the windscreen, an array of automated HMI luminaires (five 2500 watt and four 575 watt) was chosen. However, these luminaires were trained on a cylindrical diffuser which surrounded the sun simulator and the focal point. Each luminaire could be independently aimed, focused and color-selected from 16 different colors in its scroller. This allowed a wide range in the simulation capability of sky/cloud conditions.

Operational efficiency of this simulator is dramatic when compared to the "integrating sphere" concepts. At 19 kilowatts in the maximum lighting mode (sky/cloud and solar simulator luminaires all on), power requirements are still less than 1/3 of the 60 kilowatt estimated requirement for the internally illuminated concept and approximately 1/10 of the 168 kilowatts estimated that the externally illuminated concept would have required.

Lightning Flash Simulator

The lightning flash simulator system was developed by a company which specialized in special effects for the entertainment industry (and, incidentally, was also nominated for an Academy Award that year for their development of special effects). The flash unit is composed of an array of different wattage strobe lights which are used to simulate the bright first flash and the trailing sequence of flashes. Flash duration and intensity can be adjusted to obtain a multiple flash which very accurately simulates lightning activity.

Materials testing

Materials for the various elements of the ACALSS had to be evaluated to determine their suitability. Materials that are commonly used in the movie/TV industry to handle light were used for this purpose. For the ellipsoid, a highly reflective surface, yet not a mirror surface, was desirable. For this purpose, an almost specular brushed aluminum material was selected. The ellipsoid was constructed in sections from molded fiberglass. The reflective material was cut into strips and applied much like wallpaper to the interior of the fiberglass ellipsoid.

For the diffuser which surrounded the solar simulator, a material that diffused the light sources, yet transmitted most of the light, was desired. Several materials were found to be acceptable for this purpose, ranging from a cloth fabric base to a plastic base. A slit in the cylindrical diffuser allowed the light from the solar source to pass undisturbed.

Color gels for the automated scrollers were chosen to simulate the range of sky colors from sunrise to noon (sixteen choices for each of the scrollers), with several selected gels having diffusing qualities as well. Those colors with diffusers were chosen to simulate the lower light levels and colors closer to twilight.

VALIDATION AGAINST DESIGN CRITERIA

After the ACALSS was constructed, calibration and verification began. Included were calibration of the aim points of the luminaires for maximum luminance in the FFOV and positioning and aiming the smaller over-the-shoulder reflective panels for maximum illumination of the cockpit instrument panel.

Laser Alignment System

Even when the HMI luminaires were focused to full spot, finding the center of the solar simulator beam and determining its aim point was a time-consuming task. Therefore, a laser was obtained to aid in calibration and setup. The laser was mounted on the front center of the luminaire and was used in fine pointing the luminaire for the FFOV and for adjusting the small ellipsoidal panels to reflect light over-the-shoulder to the desired point on the instrument panel.

Sky/Cloud Luminance

Since the vertical angle-of-view, as seen by the pilot, decreases considerably toward his right (out the co-pilot's side of the windscreen), and in order to present a continuous set of data horizontally, luminance data shown here were taken at three elevation angles with respect to the PERP: level with the PERP, five degrees up, and eight degrees down. Luminance measurements were made with a portable luminance meter which was tripod-mounted and positioned at the PERP. As can be seen in Figure 6, the FFOV luminance peaks at about 18,000 footlamberts for the five degree up and sun almost straight ahead case. The surrounding sky luminance averages around 3,000 footlamberts, well above the design goal of 2,000. By adjusting the sun focus toward flood and adding a diffuser in front, the luminance can be increased to 10,000 footlamberts over a large area of the sky, satisfying the design goal of simulating a diffuse cloudy sky. Darkness (at the level desired for display evaluation) was accomplished by simply closing the doors behind the ADEC and turning off all the luminaires.

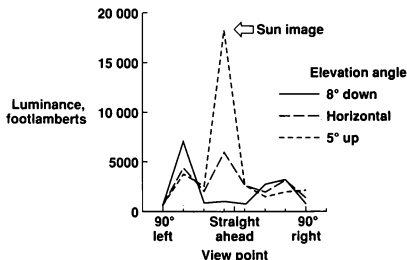


Figure 6. Validation of Sky/Cloud Condition with the Sun in the FFOV

Cockpit Illumination

Illumination of the cockpit instrument panel is highly dependent upon the windscreen and fuselage geometry. Therefore, the design goal was 10,000 footcandles at the side window. As seen from Table 2, this goal was exceeded in each of the three over-the-shoulder cases shown, with resulting illumination at the PIRP (measured with an illuminance meter having a cosine receptor) of over 8,000 footcandles. In an effort to closer simulate illumination levels experienced in a bubble cockpit, the simulated sun was aimed directly down into the front windscreen and one of the sky/cloud luminaires was directly aimed into the side windscreen. A level of over 12,000 footcandles was measured under these conditions.

Over-the-shoulder reflector panel number	Illumination at side window, footcandles		Illumination at instrument panel, footcandles
	Goal	Measured	Measured
1	10,000	21,300	5,810
2	10,000	12,000	8,120
3	10,000	16,900	6,270

Table 2. Illumination of Cockpit from Various Reflector Panel "Sun Positions"

First Completed Study

A study which was recently conducted in the ACALSS evaluated pilot/vehicle performance, under varying ambient lighting conditions, of three monochrome display devices. A laboratory-class cathode ray tube, a standard electroluminescent flat-panel display, and an enhanced-brightness electroluminescent flat-panel display were configured as a primary flight display. Subjects were asked to maintain airspeed, heading, and altitude in the presence of wind gusts as the illumination level was increased. Their airspeed error performance as a function of illumination level is shown in Figure 7.

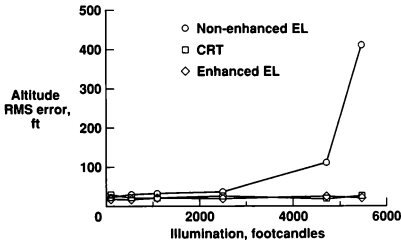


Figure 7. Variation in Display Effects Upon Performance for Three

CONCLUDING REMARKS

An aircraft cockpit ambient lighting simulator, the ACALSS, which surrounds a wide-body part-task simulator cockpit, has been constructed and validated against original design goals. These design goals, which were satisfied or exceeded, included the considered worst-case ambient lighting conditions for display viewability by a pilot. Specifically, these conditions were:

1. Sunlight illuminating the side window of the cockpit (goal: 10,000 footcandles; measured: over 12,000 footcandles).
2. A diffuse cloudy sky at a luminance of 10,000 footlamberts (exceeded over large sky area).
3. A bright sky with the sun in the FFOV (sky goal: 2,000 footlamberts; measured: 3,000 footlamberts; sun measured at 18,000 footlamberts).
4. Darkness, and
5. Lightning flashes (at a level sufficient to cause a pilot to lose his dark-adapted vision) during darkness.

Not only were the original design goals exceeded, but the final design also represented a significant increase in operational efficiency over early concepts, in that power requirements were less than 1/3 that for the best previous design.

A study has now been completed which used the over-the-shoulder illumination capability of the ACALSS in evaluating pilot/vehicle performance with three different display devices.

REFERENCES

1. Silverstein, Louis D.; Merrifield, Robin M.; Smith, Wayne D.; and Hoerner, Fredrick C.: A Systematic Program for the Development and Evaluation of Airborne Color Display Systems. Sixth Advanced Aircrew Display Symposium, May 1984.
2. Prince, David M.: Integrated Displays and Controls Design Factors for the 1990's Transport Aircraft. IEEE NAECON 80CH1554-5, 1980.
3. Hatfield, Jack J.: Advanced Electronic Displays and Their Potential in Future Transport Aircraft. IATA Technical Symposium. Aviation Technology in the 80's, Paper SYMP.81/WP34, December 1981.
4. Hatfield, Jack J.; Robertson, James B.; and Batson, Vernon M.: Advanced Crew Station Concepts, Displays, and Input/Output Technology for Civil Aircraft of the Future. IEEE/AIAA 3rd DASC, November 1979, Paper 95.
5. Caldwell, J. Robert: Airborne Electronic Color Displays - A Review of UK Activity Since 1981. Sixth Advanced Aircrew Display Symposium, May 1984.
6. Jacobsen, Alan R.: Color Liquid Crystal Displays on the Flight Deck: Human Engineering Considerations. AIAA/IEEE 8th Digital Avionics Systems Conference, October 1988.
7. Snyder, Harry L., Ph.D.: Electronic Displays: Their Strengths and Weaknesses for Advanced High Performance Aircraft. AGARD CP-371, May 1984, Paper 12.
8. Brindle, James, et al.: Flat Panel Display Technology Overview. AGARD AG-255, October 1980, Paper 2.
9. Electronic Displays for Transport Aircraft. Avionics, vol. 12, no. 3, March 1988.
10. Photometric Measurements Taken in Flight on May 17, 1972. English Translation of Thomson-CSF, Electron Tube Division (St. Egreve, France), Document No. TD/1.1423172, June 1972.
11. Crew Station Design Facility Feasibility Study. AFFDL-TR-79-3037, May 1979.
12. Wyszecki, Gunter; and Stiles, W. S.: Color Science: Concepts and Methods, Quantitative Data and Formulae. Second ed. John Wiley and Sons, 1982.
13. Monty, Robert W.; Old, Joe; and Snyder, Harry L.: Human Factors Studies of Control Configurations for Advanced Transport Aircraft. NASA Research Grant NAG-1-491, August 1985.

CORRELATING VISUAL IMAGERY WITH THE SIMULATED MISSION

Ron Matusof and Steve Schwalm – Staff Engineers
Link Flight Simulation Division of CAE-Link Corporation
Binghamton, NY

Barry A. Hicks – Systems Analyst
Link Flight Simulation Division of CAE-Link Corporation
Advanced Products Operation
Sunnyvale, CA

Abstract

In the last several years, simulation technology has advanced to a point where it is now common to talk about networking simulators and rehearsing for specific missions using real-world terrain, weather, and intelligence data. The majority of these advances have come in the area of visual generation and display. As a result of the recent interest in mission rehearsal and simulator networking, there has been a concerted effort to produce visual imagery which can be properly correlated between the out-the-window scenes and the various sensors within a given simulator, as well as between networked simulators. While this type of effort is extremely important, it overlooks a crucial aspect of training the modern crew: correlation of the simulated mission with the visual imagery. The problem facing the training system developer arises from the fact that, in general, the visual system and the training system are developed independently without regard to how one system will correlate with the other. This paper discusses this problem and one solution known as environmental interrogation. It presents the requirements for environmental interrogation, surveys current methods to extract environmental information to correlate with the visual imagery, and proposes a method of evaluating the environmental interrogation requirements for a mission simulation.

Introduction

Major advances in the capabilities and fidelity of visual generation and display systems have occurred over the last decade. These advances parallel changes in training concepts which are rapidly changing the way in which simulation is used to train crews. Concepts such as simulator networking for team training are not only possible, but are currently in operation, albeit in prototype systems.^{1,2} The recent interest in simulator networking and mission rehearsal has increased interest in producing visual imagery which can be properly correlated between all sensors within a given simulator and between networked simulators. The joint services Project 2851, for example, is scheduled to

provide a common visual/sensor database to be used by all visual contractors working on military contracts. This type of effort is important since it will provide a common visual gaming area in which combat teams, training within a network of dissimilar simulators, can perform a mission. However, a more fundamental problem which is being largely overlooked involves correlation of the simulated mission with the visual imagery, which is a crucial aspect of training or rehearsal for any crew. This correlation problem applies equally to networked simulators and to dedicated, stand-alone simulations.

The correlation of visual imagery and the simulated mission takes on added significance as the skill level of the user increases. For high-gain tasks such as combat mission team training, low-level or nap-of-the-earth (NOE) flight, and mission rehearsal, the proper coordination of resources and employment of equipment is highly dependent upon how well the overall system fidelity is portrayed. What the trainees expect to see as a result of their interaction with the simulated environment is a function of their collective experience. Failure to match their expectations may result in reduced transfer of training from the training system to the real world. The training system developer is faced with the problem of integrating a visual system and a training system which have been developed independently without regard to how one system will correlate with other.

One solution is environmental interrogation. Environmental interrogation refers to the processing which provides information concerning the interactions between any two points in the simulated environment. In this sense, environmental interrogation provides more than occulting or collision detection. It allows interactions between the ownship and the environment, as well as interactions between the elements of the environment which are independent of the ownship but nonetheless of training significance.

In today's military simulations, the interaction with the environment is a required feature. The interactions between the ownship and the ground, trees, and other objects within the environment be-

come increasingly important as the altitudes at which aircraft fly decrease into NOE. Additionally, for high-fidelity threat or weather portrayals, the complexity of interactions between the ownship and the environment, or between the elements of the environment, is markedly increased for the NOE profile. This increase in complexity demands that the simulation host computer determine conditions such as instantaneous height above terrain, crash detection, range to objects, and other forms of information which may be obtained only by interrogating the simulated environment.

Interrogating the Environment

The basic concepts of environmental interrogation are by no means new. Even early simulators defined a ground plane at an altitude of zero feet (relative to mean sea level) and caused a "crash" condition whenever an aircraft altitude dropped below the ground plane. For this case, the world was considered to be a perfectly flat surface. Very few real-world missions, however, occur over perfectly flat surfaces, and this fact has led to the use of lookup tables to provide environmental information.

The lookup table increases the fidelity of ownship/environment interactions by providing accurate altitude above terrain and crash determinations. However, lookup tables are computationally intensive and therefore usually model only a small area of the environment, such as an airfield. The accuracy provided by the lookup tables is directly related to the number of samples stored within the table. Additionally, the use of lookup tables becomes extremely difficult if newer visual concepts, such as dynamic terrain,³ are introduced, since the number of lookup tables required is related to both the amount of terrain which can be modified and the speed with which lookups can be performed.

A modern technique for environmental interrogation, first developed on the CAE-Link Army Tactical Digital Image Generator (ATACDIG), is known as feature correlation. Feature correlation defines the information required for environmental correlation by defining a set of three-dimensional geometric volumes. Each volume consists of six planes: the near, far, top, bottom, left, and right planes. Each plane is defined in terms of an offset distance and rotation angle from its origin. Yaw angles (rotation around the Z axis) may be applied to the left and right planes, while pitch angles (rotation around the Y axis) may be applied to the remaining four planes (see Figure 1).

The requests determine not only if an occult or crash has occurred, but the range to the occult/crash as well as the type of object intersected. This additional information is important since different

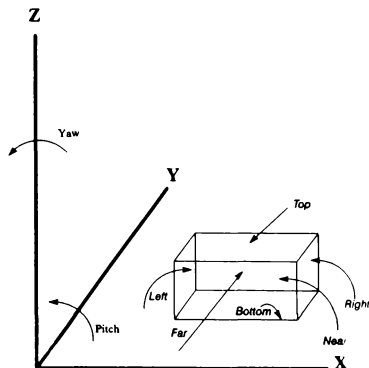


Figure 1 Feature Correlation Volume

types of objects will be perceived as having unique effects upon the ownship. For example, a helicopter bumping its fuselage into a reinforced cement structure will most likely crash. An identical bump into a pine tree will most likely cause a jolt to the aircraft, but will not result in a destructive crash. Feature correlation also provides the ability to specify which classes of visual entities should be excluded from a given feature correlation test. Smoke and dust, for example, are often excluded from ownship crash tests while narrow objects, such as telephone poles are typically excluded from altimeter tests. Exclusions are also important for the simulation of threat line-of-sight, which may be occulted by smoke for optical tracking systems but may not be occulted for radar guided systems.

Several basic types of feature correlation test volumes are shown in Figure 2a-c. The volume test (Figure 2a) consists of an enclosed volume which, in general, covers all or part of the ownship. These volumes are very useful in making crash (contact) determinations between the ownship and the environment. While both the shape and size of the volume are known for volume tests, it is often useful to obtain information about an area whose exact size is unknown. For these situations, three other test types are defined. The altimeter test (Figure 2b) and the rangefinder test (Figure 2c) consist of pyramid-shaped windows whose ranges vary from short distances (around 500 meters) to relatively large distances (on the order of the retrieval range

information needed to correlate the environment with the simulated mission.

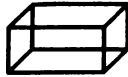


Figure 2a Volume Test

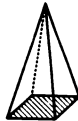


Figure 2b Altimeter Test



Figure 2c Rangefinder Test

of the image generator). Although conic-shaped windows provide more accurate information for correlation with real-world applications, they require computationally intensive implementations. Pyramids provide acceptable resolution with reduced computational complexity. Altimeter tests determine which objects intersect the entire volume, while rangefinder tests determine which objects intersect the line of sight along the upper left-hand corner of the corresponding window. Occult requests are identical to rangefinder requests except that they do not distinguish between the types of objects which have caused the occult.

For the missions which are nearest to the ground, and therefore require the highest fidelity of environmental interrogation, there are six types of

Altitude Measurement

Altitude can be defined as height above ground level (AGL), height above average terrain (HAAT), or radar altitude. Each altitude is calculated independently of the others and has unique implications for the mission. Radar altitude, for example, is referenced to the aircraft body frame and is often used by helicopter pilots to determine the closure rate of the aircraft to a large terrain feature. AGL is unsuitable for this function, since it is always measured perpendicular to the earth frame of reference. However, the fact that AGL is measured in this way makes it a valuable tool in correlating the interactions between an observer referenced in the earth frame and the ownship referenced in the aircraft body frame. Both AGL and radar altitude provide a relationship between the terrain and the ownship in a specific direction. However, it is often useful to be able to correlate the general impression of the terrain with the ownship for calculations involving radar or radio attenuation, as well as weather effects. This general impression of the terrain is often expressed in terms of HAAT. Figure 3 illustrates the differences between these three altitude measurements.

Several methods exist for measuring altitude. The simplest method involves the use of a lookup table of terrain altitudes based upon latitude and longitude. A variation of this approach interpolates between the surrounding sample points to provide a more accurate approximation. The accuracy of either approach is limited by the density of the sampling space used. As the sampling density increases, so does the memory required to store the same amount of area. Some systems use varying densities based on the complexity of a given area within the environment to reduce the memory requirements.

Feature correlation can be used to measure altitude by defining either an altimeter test (if radar altitude is required) or a rangefinder request (if AGL is required) and originating the request at the known altitude (above mean sea level) of the ownship. The returned distance to the occulting object represents the measured altitude.

Terrain Relief

Often, the general impression of the terrain provided by HAAT is insufficient for mission tasks. Terrain relief provides the information necessary to correlate the placement and movement of objects relative to the simulated terrain. For example, placing targets on the ground requires knowing the attitude of the terrain at the specific location of the target.

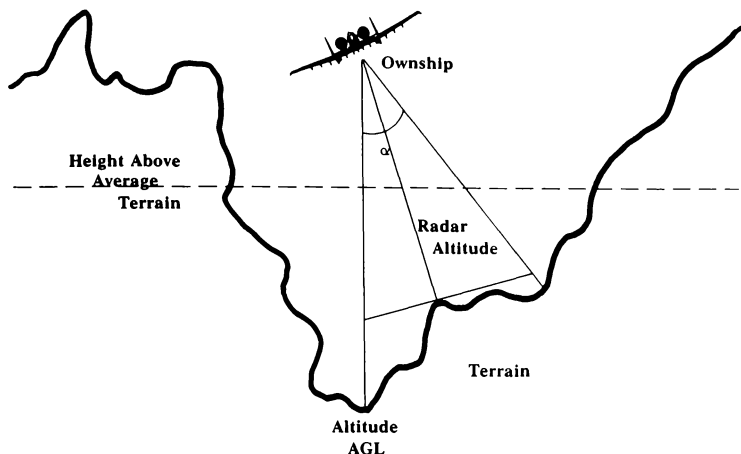


Figure 3 Methods of Altitude Measurement

For target movement, it is often useful to know the attitude of the terrain in the general vicinity of the target so that all motion will be smooth and intersection of the target with the database does not occur.

The terrain relief represents information which is generally required by the image generator to produce the visual scene. An attitude matrix, representing the rotation of the plane on which a specified point lies can be calculated easily and provided to the host computer for target placement.

Terrain Object Annotation⁴

Many times, knowing the relief of the terrain still does not provide sufficient information to correlate the actions of the ownship with the visual imagery. Terrain object annotation determines at any instant the type of terrain (water, ice, dry ground, etc.) at a particular location. This information is very important for ground effect simulations of aircraft, as well as target movement and engagement simulations.

The information required for terrain object annotation can be supplied by using lookup tables. However, for NOE profiles correlation with the visual image requires a large number of samples. Once again, computational resources determine the accuracy and fidelity of the terrain object annotations.

By their very nature, feature correlation tests (except for occulting tests) provide terrain object annotations. The ability to exclude classes of visual entities provides an additional advantage since the information required from the terrain object annotation may not include tactically insignificant objects such as telephone poles or small rocks.

Occulting

This basic feature provides a boolean indication of whether two points maintain a mathematical line of sight (LOS). This feature can be used for basic threat and radio simulations where attenuation of signals is not important.

Rangefinding

The rangefinder feature is used to calculate the distance between two objects in space, measured along the line of sight. On some systems, these requests are limited to having one endpoint at the eyepoint position, while on others, both points can be independent of the ownship. The rangefinding feature derives its name from the simulation of a laser rangefinder. In this case, only the origin of the beam is known and the location of the laser spot must be determined by calculating the range to the first intersection along the attitude of the laser beam. The rangefinder request determines not only that an intersection has occurred, but where the intersection occurred and what object was intersected. This information has applications for attenuation of radio, radar, and laser transmissions.

Collision Detection

This feature is used to determine body-on-body collisions of objects within the visual data base. These determinations are used for such diverse purposes as ownship fuselage crash, fuselage "bump" caused by intersection with a nonrigid object such as a tree, and weapon impacts.

There are a number of ways to determine collision detection. The simplest approach involves using the aircraft's height above the terrain. With this method, the altitude of the ownship is compared to the altitude of the terrain. Crash occurs when the difference between the two altitudes falls below a predetermined threshold.

Another method of collision detection mathematically determines if two objects occupy the same geometric location. There are three different approaches to point collision detection.

The first uses spherical objects defined around points of interest which approximate objects being tested for collision. If the spheres around each point intersect, then a collision has occurred. The second method involves defining nonspherical volumes around objects of interest and checking to see if the ownship falls within any of the volumes. A variation of the point-sphere detection uses vectors rather than single points. The vector-sphere collision detection method provides greater flexibility and accuracy, but at a cost of computational time. All three methods, and permutations of them, have been successfully demonstrated to meet very demanding collision detection requirements.⁵

Feature correlation defines a crash volume which approximates the size of the ownship. Objects within the environment are checked for intersection with the crash volume. A crash condition occurs when any object (other than those in an entity class excluded from the test) intersects the crash volume. A crash type indicating "hard" crash (i.e., a destructive crash, which occurs when an aircraft flies into a mountain) or a "soft" crash (e.g., an intersection with objects such as vegetation, which will not cause a destructive crash) is provided so that appropriate aural, motion, and aerodynamic effects can be simulated to correlate with the visual imagery.

Correlation Issues

As stated above, the correlation of the environmental interrogation database is very important. Like any simulation, the accuracy of the correlation depends on the types of tasks being trained or analyzed. These correlation requirements vary greatly for both the terrain and cultural features based on the task at hand.

The correlation of altitude requests to the environment varies for the simulated function. A radar altitude requires only the resolution that can be displayed on the real-world indicator, which is usually one foot. The resolution for performing a sloped landing is much higher since the person flying the vehicle can judge his altitude through the visual system. The correlation for positioning of a target model varies based on its proximity to the ground. As the model approaches the ground, the target must be placed more accurately. This becomes crucial in some visual systems since their occulting algorithms do not support the target penetrating the ground faces.

For line of sight, the correlation to the visual system is also based on the request type. For a radio line-of-sight request, a resolution of one meter is more than sufficient. This is also the case for radar line of sight calculations. Visual line-of-sight is a different case.

The visual line of sight is heavily dependent on the cultural features within the environment. If the position of a cultural entity displayed within the two systems differs by a small amount, the angular error induced can be extremely significant to the simulation.

These correlation problems are similar to the one currently being addressed by Project 2851 for visual and sensor systems. If correlation can be achieved between these systems, then it can be assumed that correlation can be achieved for the environmental interrogation systems. This means that the burden of performing environmental interrogation functions can be removed from the visual systems, freeing them to perform their primary task of generating images.

Recommendations

All missions are not the same, and therefore the required fidelity of interactions between the simulated environment and the ownship varies between simulations. Determining the types, amount, and fidelity of environmental interrogation required for a particular training system involves considering the training objectives for the simulation, the technical impact of providing varying degrees of environmental interrogation, and the associated costs. As is the case with most designs, trade-offs will have to be made in order to meet these challenging and often conflicting considerations.

The training objectives of a simulation generally relate to the mission of the vehicle being simulated. In order to properly perform a mission, a crew requires a set of information which relates actions of the environment to reactions of the ownship, and vice versa. The degree with which the visual simulation must correlate with the actions of the environment is often determined subjectively. However, the basic requirements can be

determined objectively by identifying which interactions may be observed by an individual crewmember.

Once all interactions have been identified, the concept of training value must be applied to eliminate interactions which provide no positive benefit for the student. The list of interactions which have training value will vary between training systems. For example, radio line-of-sight determinations may have a great deal of significance to a fighter training system, since the interaction between simulated vehicles affects the command and control structure of the threat. The command and control structure of the threat may affect the information perceived, and ultimately the decisions made by the fighter crew. Less than adequate simulation may lead to negative training.

Three distinct attributes about environmental/ownship interactions must be determined in order to properly specify an environmental interrogation system:

- 1) Types of Tests. All six types of environmental interrogation (altitude, terrain relief, terrain object annotation, occulting, rangefinding, and collision detection) are often not required for specific simulation applications. There is little value in providing environmental interrogation capabilities if the interrogation produces information which cannot normally be perceived by the crew. A commercial airline simulator, in general, has little use for terrain relief information, since the simulated mission is always supposed to end on a level runway (that is, the terrain relief of the areas where the ownship interacts with terrain is always known). Fidelity of the simulation also affects the types of tests required. A part-task trainer whose primary purpose is to train radar intercept operations may not require collision detection, since collisions do not prevent achieving the training objectives of simulations.
- 2) Fidelity of Tests. The fidelity of the tests is a function of the training system specification and the accuracy of the simulator software which uses the information from the environmental interrogation. Many training system specifications are now identifying maximum tolerances for collision detection. A commonly specified tolerance requires detection of collisions within five feet of any part of the ownship fuselage. There have been some recent specifications which also state requirements for the accuracy of other types of tests, such as altitude and line-of-sight determinations. When the accuracy of

the environmental interrogation is not specified by a customer, the requirements can be determined by examining the systems which will use the information provided by environmental interrogation. The accuracy of these systems will determine the required fidelity of the environmental interrogation. There is little benefit to providing information accurate to a millimeter when the distances used by the simulated system are measured in kilometers.

- 3) Required Frequency. An often overlooked factor when evaluating environmental interrogation requirements is the minimum frequency at which interactions between the simulated environment and the ownship must occur to avoid degradation of the training system. The required frequency is a function of several factors, most notably maximum ownship velocity. For example, an aircraft moving at 1,000 knots traverses slightly more than 514 meters in a second. If it is desired to detect crashes into objects at least 5 meters in size, then crash detection must be provided at a minimum rate of 103 times per second. A second factor which should be considered is the simulated reaction time of the environment. For example, if a simulated threat "thinks" five times per second (that is, the software simulation of the threat logic is updated five times per second), then his perception of the environment (lines of sight to the ownship and other targets) must be updated at this rate. Updating perceptions at any rate greater than the iteration rate of the software provides no additional fidelity.

By evaluating the information required by a crew to interact with the environment, it is relatively easy to determine the environmental interrogation requirements. Unfortunately, the requirements for environmental interrogation are often larger than the technical capabilities of the training system or the available budget of the customer. The customer must then conduct a cost trade-off and compare the effects on training objectives against the cost benefit of providing less capability for environmental interrogation.

About the Authors

Barry A. Hicks is a Systems Analyst with CAE-Link Corporation in Sunnyvale, California. Mr. Hicks has 5 years of experience developing and integrating real-time software for Link's Digital Image Generation systems. He is technical lead engineer on the Special Operations Aviation Combat Mission Simulation development team, and is currently

involved in design and analysis for the AH-64 Apache Combat Mission Simulator Block Update program. Mr. Hicks holds a Bachelor of Science in Computer Science from the University of California at Davis. He is currently pursuing a Master of Science degree in Computer Engineering at Santa Clara University.

Steven Schwalm is a Staff Engineer with CAE-Link in Binghamton, New York. Over the past 7 years, Mr. Schwalm has been developing image generator real-time, database, and integration software. He is currently the lead engineer for the MULTISIM simulator networking IR&D Program. He also acts as a consulting engineer for the AH-64 Combat Mission Simulator Block Update and LH simulator development programs. He holds a Bachelor of Science in Computer Science from the Pennsylvania State University.

Ron Matusof is a Staff Engineer with CAE-Link Corporation in Binghamton, New York. For the last 7 years he has worked on the simulation of avionics, tactical systems, and electronic warfare systems for a variety of military training devices. He is currently the Principal Investigator for Mission Generation and Rehearsal IR&D and is involved in tactical system simulation for both the Special Operations Aviation Combat Mission Simulator and the MULTISIM simulator network. He holds a Bachelor

of Science in Electrical Engineering from the University of Pittsburgh.

References

1. Dees, J.W., and Cornett, T.R., "Simulator Networking in Helicopter Air-to-Air Combat Training", AIAA Flight Simulation Technologies Conference, Boston, MA, September 1989.
2. Thorpe, J.A., "War Fighting with SIMNET - A Report from the Front", Interservice/Industry Training System Conference, Orlando, FL, December 1988.
3. Moshell, J.M., Hughes, C.E., Goldiez, B., Blau, B., Li, X., "Dynamic Terrain in Networked Visual Simulations," The Second Annual Conference on Standards and Interoperability of Defense Simulations, Orlando, FL, January 1990.
4. Wyckoff, B.H., "Managing Cost/Performance Tradeoffs for Successful Visual Training", Interservice/Industry Training Systems Conference, Fort Worth, TX, November 1989.
5. Ciaponi, L.J., "Digital Simulation of a 6-DOF Multi-Body Dispense With Body-on-Body Collision Detection", NAECON, Dayton, OH, May 1989.

REAL-TIME SCENE GENERATOR FOR TESTING OPTICAL SEEKERS

William C. Wagner
Rockwell International, Space Systems Division
Downey, California

Abstract

The signal inject scene generator (SISG) is a real-time image generation system used to create a simulated focal plane array output and digitally inject it into an optical seeker under test. Since the injected signal is used in lieu of the real focal plane electronics to test the seeker's signal and data processing algorithms, the distortions and characteristics of the optics and focal plane physics are modeled and included. The initial requirements were driven by the testing of a four-band infrared (IR) seeker with exceptionally high update rates. However, future plans for testing other seeker-focal plane configurations drove the design to a generic, reconfigurable, scalable scene generator system.

Introduction

The current research in the use of kinetic vehicles (KVs) for missile or satellite defense has increased the need for advanced optical seekers to guide the vehicles. In the KV concept, the kinetic vehicle destroys another vehicle by the sheer transfer of kinetic energy to the target vehicle. The typical KV mission is an exoatmospheric engagement in which the closing speed between the KV and target is many miles per second—the concept is much like hitting one bullet with another. KVs require high-cycle-rate subsystems that can quickly respond to last-moment corrections to target vehicle maneuvers. The seeker subsystem provides the precise steering information required to intercept a target.

The seeker typically comprises an optical system that collects and focuses incoming images onto an array of photodetectors arranged in a planar matrix called the focal plane array (FPA). The detectors are sensitive to one or more specific wavebands in the electromagnetic spectrum, either IR, ultraviolet (UV), visible, or some combination. The resulting energy that impinges on a detector registers as an intensity value, and the array's output is essentially a digitized picture of intensity levels for the specific waveband of the detectors. This image is then sent to the seeker's processing electronics, which analyze the image and perform target tracking, aimpoint selection, and steering.

The testing of advanced optical seekers in a real-time, closed-loop simulation environment requires the generation of highly accurate scenes of a wide variety of possible targets against expected backgrounds. These seekers may have extremely high refresh rates, which makes conventional commercial methods of

scene generation inadequate. Presenting an image to the seeker under test can also pose serious technical and financial problems when the spectral bands are in the IR frequencies. For test scenarios that require highly dynamic and extended target images, solutions can cost upwards of \$10 million. If verifying the performance of seeker algorithms is the main goal of testing, an alternate approach can be taken to lower cost and risk—signal injection. In this method, the optics and focal plane hardware are bypassed; therefore, not only is the target/background scene generated, but the seeker's motion, optical effects, and focal plane characteristics are also applied. The resulting blurred-picture mosaic simulates the FPA output and is electronically substituted for the real FPA. Fortunately, because the signal-processing algorithms are independent of the optics and focal plane assembly, the performance of the remaining downstream seeker electronics can be accurately assessed by using the injected data stream. The challenge, then, is to reliably and efficiently create focal plane images that stimulate the seeker's signal processor, which represents a significant portion of the seeker system testing.

The SISG project was initiated to develop the capability to test advanced seeker brassboards in a simulated closed-loop environment. Fig. 1 shows the relationship of elements in the hardware-in-the-loop simulation.

The development of the basic approach and algorithm design was driven by the testing requirements of the four-color IR seeker used in Rockwell's kinetic kill vehicle (KKV) concept for the Space-Based Interceptor (SBI) program. Therefore, this paper refers predominantly to that configuration. However, since this development was a capital investment by Rockwell, every attempt was made to design the scene generator to address the potential needs of other types and sizes of seekers. This led to a generic and modular design approach that uses data to configure the characteristics of the scene generator to support specific programs.

Requirements

The SBI mission scenarios present a boost or postboost target vehicle against a variety of backgrounds (ground, cloud, earth limb, space, etc.) and under a high-closing-rate, hit-to-kill engagement. The target scenes range from far-field point sources to near-field extended images that can overfill the field of view (FOV). Since the target may be a missile hardbody in the presence of its exhaust plume, the requirement of variable transparency of the overall

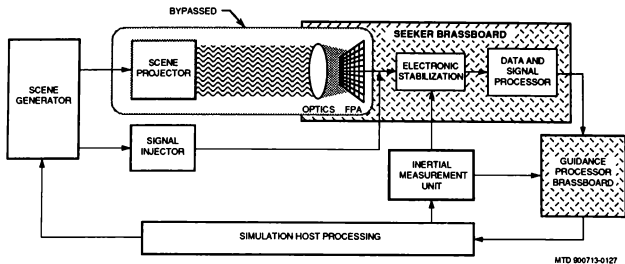


Fig. 1. Hardware-in-the-loop simulation block diagram.

target image over the background becomes a design consideration. The background may "bleed" through some parts of the plume and be totally blocked by the hardbody. Since the seeker is body mounted, any vibrations are directly transmitted to the seeker and result in high-speed image motion on the focal plane. This motion is compensated for in the seeker's design by electronically stabilizing the focal plane, so proper dynamic image placement is essential. The seeker's processor also contains image-processing algorithms that derive images of subpixel resolution from pixel-size focal plane elements. This requires that the scene generator's positional accuracy and feature granularity be the same as or less than the subpixel resolution. For the SBI seeker, there are 16 subpixels per focal plane pixel or a 4 times focal plane element accuracy increase per side. Two further design considerations are the focal plane spatial configuration and refresh rate. The SBI seeker is a $2n \times 2n$ array in which one fourth of the elements (pixels) are sensitive to one of the four IR bands or colors. Each set of color pixels is in an $n \times n$ configuration and is spatially interleaved with the other bands (see Fig. 2). What results is an $n \times n$ array that takes up the area of a $(2n-1) \times (2n-1)$ array, and each pixel, for any one color, is surrounded by an inactive pixel (in that color) in all directions. This

configuration essentially requires images to be generated that have a scene content of $8n \times 8n$ elements per color.

The SBI seeker's electronics are designed to read the focal plane at over 3,200 Hz. Updating the scene at that rate would require enormous computational requirements, so it was decided early in the program that a 400-Hz scene update rate would be an acceptable compromise. However, an open-loop mode that would update the scenes at the higher rate was required. This provides a means to drive the seeker at the high rate to accurately test electronic stabilization and serves as a way to assess the effect of the 400-Hz compromise. This mode consisted of prebuilding the images per a prerun engagement scenario and then playing them back in the proper sequence at the high rate. This eliminated the high on-line computational requirements, yet did define the memory and data throughput upper limits.

In order to bound the problem even further and enable an efficient design to emerge, several assumptions were proposed and agreed to by the program. First, the short-duration, high-closing-rate, hit-to-kill engagement scenarios typical of KKV's result in the aspect of the target image (with respect to the seeker) being relatively constant over the engagement. This meant that for a particular engagement only one target image configuration (head-on, side, three-quarter view, etc.) would be viewed by the seeker. This eliminated any dynamic three-dimensional (3-D) image manipulation requirements and allowed the use of two-dimensional (2-D) bit-mapped images to be used. This greatly simplified the on-line computations. Another simplifying assumption concerned the seeker's roll axis engagement dynamics. A guidance requirement imposed by the program was to keep the roll attitude of the vehicle constant within a tight deadband over the engagement. This aided the image-processing tasks of the seeker algorithms. It also meant that the scene generator need not dynamically roll the image, which significantly reduced on-line computational requirements.

The data bases that make up the basic object space target and background images are derived from either measured or computed data called phenomenology-based data. Proven methods of creating these data bases are in place and are not a concern in this paper. The data bases are to be assumed as external inputs to the design. Each raw phenomenology data base image is a 2-D array in which each element is assigned an intensity level corresponding to the inherent brightness of the image at a particular wavelength. At least one image is required for the target and one for the background in each of the color bands of the seeker. Since a large background area can be traversed over an engagement, the raw data base must contain a large enough area.

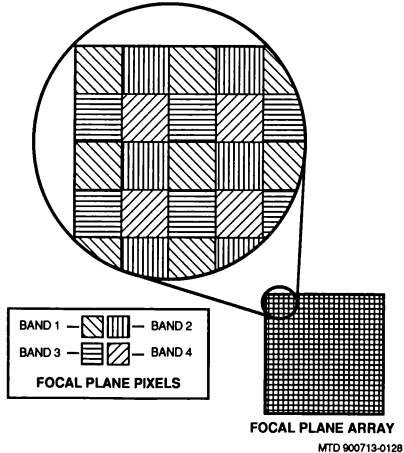


Fig. 2. SBI focal plane configuration.

Approach

An analysis of the scene generator requirements and assumptions resulted in the top-level functional overview shown in Fig. 3. Much like for a movie, each scene is assembled by superimposing objects in the foreground over a backdrop. More specifically, for each scene the instantaneous FOV is windowed over the background image; the target image is rotated (per the initial roll angle) and scaled to the instantaneous size of the target, given the FOV and the current range (the intensity is also adjusted); the target image is then overlaid on the background at the instantaneous position with respect to the FOV; and finally, the seeker optics and focal plane characterization is applied, resulting in the FPA. Of the computational requirements studied, optical convolution is by far the most time-consuming and one that it is currently not feasible to perform at 400 Hz. To resolve this, one final assumption was adopted that allowed the image data bases to be optically convolved prior to the on-line functions. It was determined that the resulting errors introduced by this method were very small given the target/background image relationships and that the seeker's image-processing algorithms were insensitive to these errors. This shifted the major processing overhead to an off-line computation, and the process of allocating scene generator functions to on-line or off-line tasks began.

The other major computationally intensive function is the scaling of the target image's size per the instantaneous range. What was proposed was to build a series of target images that are prescaled (and intensity corrected), starting from the smallest 2-by-2 subpixel image and geometrically increasing in size by a subpixel on a side to images that overfilled the FOV. The on-line system would then choose the proper image array from the on-line data base as a function of instantaneous range and then further correct for intensity variations from the reference range.

Once a basic design approach and allocation had been determined, requirements for a computer system that would handle the computational and data rate demands were formulated. It was determined that to best evaluate the available systems, a benchmark program that performs the actual worst-case computations (experienced during an engagement) would be written and then run on contending hardware platforms. In fact, the worst-case computations occur when the target image fills or overfills the FOV, so a fairly simple set of algorithms was devised to do this. What became apparent was that the on-line calculations can easily be spread over many processors operating in parallel using a very large, common, static data base. Also, integer arithmetic was all that was required during the on-line calculations.

For the system to be considered viable, the vendors had to first run and time the benchmark routine sized to a single-color full FPA. This determined processor speed and sized the number of processors potentially required to achieve the 2.2-millisecond time limit. If more than one processor was required, each of these n -processors then computes $1/n$ th of the benchmark in parallel by using the data base memory configuration proposed and storing the results in one commonly located output array. This second step revealed any potential data throughput bottlenecks. Since this was a capital investment by Rockwell, consideration of using the computer system on other projects for other purposes led to the requirement that only the multiple-instruction/multiple-data (MIMD) processor architecture be considered. However, single-instruction/multiple-data (SIMD) processors would also be capable of performing the scene generator task. Only three vendors had systems that met the benchmark, and the BBN TC2000 was chosen based on its overall technical merit.

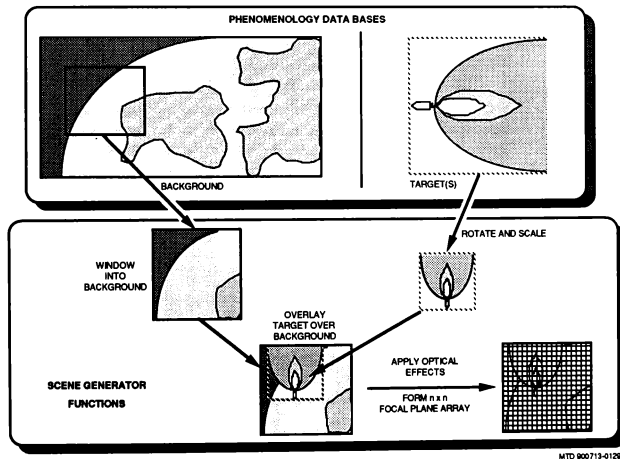


Fig. 3. Top-level scene generator functions.

Design

With the computer system selected and the purchasing cycle started, preliminary design of the off-line and on-line tasks started. Fig. 4 shows the functional block diagram of the scene generator allocation. The off-line system is designed to operate in nonreal time yet have as high a throughput as possible. Its major function is to take generalized engagement scenarios that define target types, background type, engagement aspect angle, and other specialized configurations and, from raw phenomenology data bases and sensor optical prescriptions, build an on-line data base. For the closed-loop mode, the on-line system then uses this memory-resident data base to dynamically compose each 400-Hz scene using target, background, and vehicle relative geometry data provided by the closed-loop simulation. At the refresh rate of the focal plane, the digital signal inject (DSI) interface picks up the latest generated FPA and sends it to the seeker's electronics. For the open-loop mode, the on-line data base consists of prebuilt FPAs that reflect the detailed engagement dynamics produced by a simulation using a math-modeled seeker. The on-line system simply sends these FPAs to the DSI in the right order and at the focal plane refresh rate. In addition, simulated inertial measurement unit outputs that correlate to the engagement (also prebuilt) are synchronously played back to the seeker for proper electronic stabilization.

The off-line routines are kept very generic and modular and have no direct user interface. Their commands, configuration, and data come from driver files. This keeps their design stable and uninfluenced by specifics. The user interface is the set of programs that build these driver files. Its basic purpose is to interact with the user and manipulate and format the data that go to the files. Its design can be constantly evolving to provide higher levels of user interaction or additional functions while buffering these "upgrades" from the basic application routines. The remainder of this paper deals with the design of the closed-loop mode systems only.

The basic functions of the off-line, closed-loop mode application routines are shown in Fig. 5. There are four basic functions that are progressively applied to the raw phenomenology images: rotate, scale, optics, and presum.

Rotate takes a 2-D array and rotates it about the center. This is done once per image.

Scale takes a 2-D image (in object space) and, based on a specific range value, scales it up or down to form the correct size image for a given FOV and object dimension. Scale is called many times. It builds one image per range value in the range table and uses a rotated phenomenology image associated with that range. This allows for two open-ended capabilities. First, for a given engagement, several raw images can be used—a far-field image for long ranges that is lower on detail but may contain a longer plume and a near-field image that gives more detail once the plume end is outside the FOV. Second, the sequence of range values is nominally selected to correspond to those discrete ranges where the target size increases by a subpixel. However, the table-driven approach can be used to minimize the number of images required and, hence, optimize memory. Specifically, there is a range point in an engagement prior to which a specific size target image is applicable for more than one 400-Hz frame and after which a new size image is required every frame. Prior to this point, the images must be built in geometrically increasing order since they will all be used because the image will grow less than a subpixel between frames. After this point, the image grows by one or more subpixels between frames, so only a subset of the geometric set needs to be built because many of the sizes will be skipped over. This occurs at a point in the engagement where the images are large; therefore, a considerable amount of memory storage is saved. A combination of these capabilities can be used to handle temporal-type effects. For instance, if staging were desired to occur, a series of raw data base images depicting the phenomenon could be tied to occur at a specific range point, and these images would be scaled and stored so that the on-line system would "play them back" at the proper time

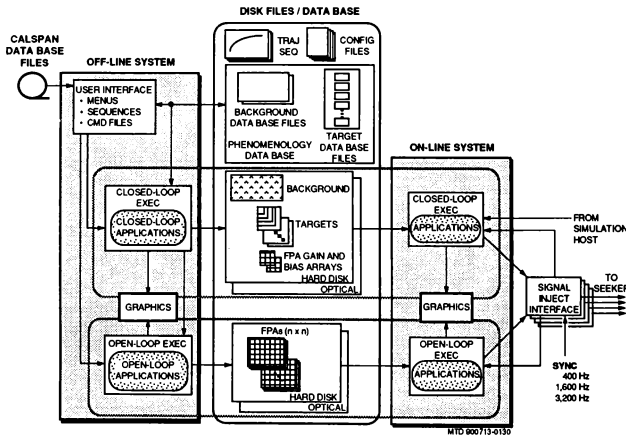


Fig. 4. Scene generator functional block diagram.

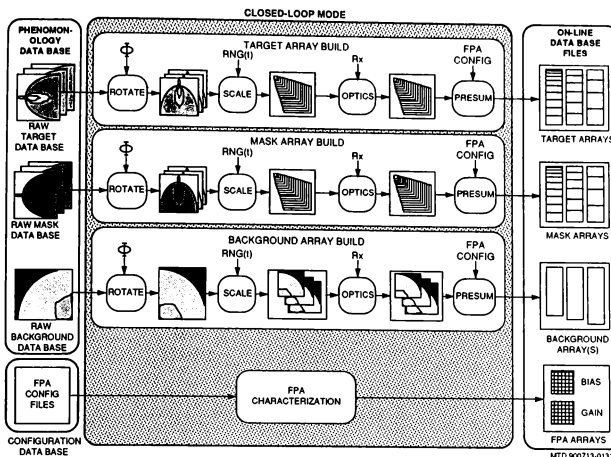


Fig. 5. Off-line system functional flow.

and sequence. The advantages of this approach are that the "intelligence" to do this is contained in the user interface and the basic off-line system is unaffected. Therefore, custom scenarios not yet thought of can be easily handled by the basic overall system design.

Optics takes each image that is produced by scale and optically convolves it based on the point spread function resulting from the seeker's optical prescription. It is generically written and is data configured.

Finally, presum takes the subpixel images and forms a series of presumed pixel arrays that take into account the inactive area on the focal plane (per color) yet still retains subpixel accuracy. The use of this technique resulted in a doubling of on-line processing speed and, hence, a halving of the number of processors required per color. The presum routine depends on the focal plane configuration and must be partially recoded for different configurations. All other routines are data driven and require no recoding for different sensors, targets, or backgrounds. These routines are also applied to the target mask images (described later) and the background arrays.

The FPA characterization function is a focal-plane-dependent function that produces a bias array and a gain array. These arrays are used to modify the final on-line pixel scenes per the characteristics of a specific focal plane or to test possible FPA degradations. Off-nominal detector gains, including dead or overactive pixels, are available.

The products of the off-line system are the on-line data base files that are stored on disk for later loading to memory and also archiving. The same parallel-processing platform that is used for the on-line system is also used to build these on-line data bases. The approach taken to make the off-line computations parallel is to use one processor to generate a task queue of scale/optics/presum groups for the target image sizes required. Then the available

processors each take an assignment from this task queue, execute it, and return to the queue for the next assignment. This coarse-grained parallelization approach minimizes common parallel-processing traps such as deadlocks, race conditions, etc., while making the configuration easy to modify.

Fig. 6 shows the functional flow of the on-line closed-loop system. Prior to a simulation run, the on-line data base arrays are loaded into memory and the following sequence is repeated during each 400-Hz frame. Based on the current range, the proper size target (and mask) array is selected; and, since it is known where the vehicle is pointing, an instantaneous "window" into the large background array is determined. Then using the relative geometry of the vehicle-target, the correct placement of the target with respect to the window is calculated. The background area under the target array is first multiplied by the mask array. This accounts for target transparency effects by allowing parts of the background to bleed through. (The mask value is a zero for totally opaque areas and a one for totally transparent sections, with variations in between.) Next, the target array is added, and the result represents the image that falls on the focal plane pixels. An on-line routine generates the characteristic focal plane noise, which is added to a static bias value. This is then added to the FPA image, and the result is multiplied by a per-pixel gain to produce the final FPA output, which is routed to the signal inject interface and optionally stored for off-line analysis.

Fig. 7 shows the hardware allocation of the on-line system. One common processor, using range and pointing data from the simulation, determines the appropriate target and background arrays to use, along with the initial and final array indices. These indices indicate where in the background array to start and stop and where to start overlaying the target over the background. These indices are transmitted to the processors (called worker nodes), each of which operates on its respective piece of the scene. For more complex or faster focal planes, more nodes can be added to achieve

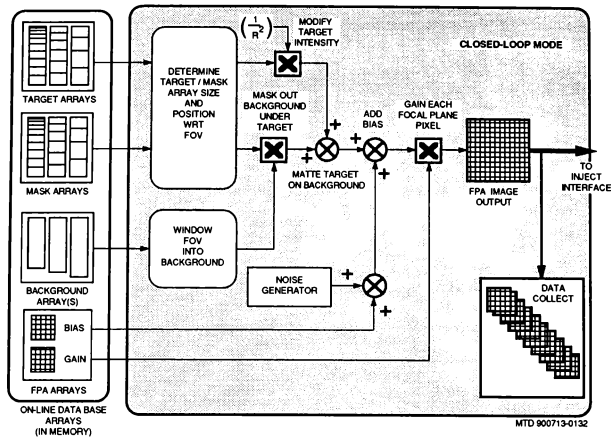


Fig. 6. On-line system functional flow.

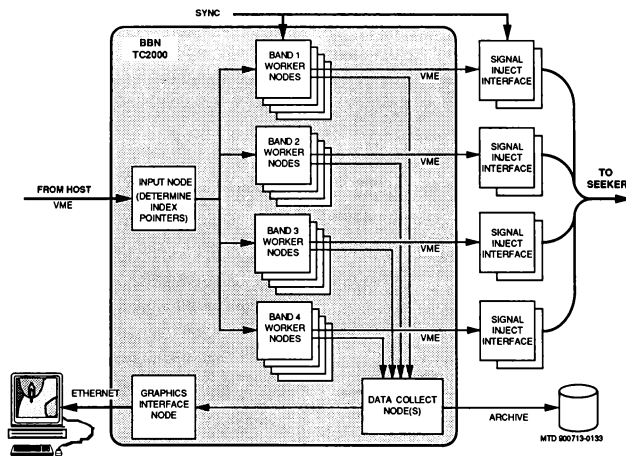


Fig. 7. On-line system hardware block diagram.

frame time; and, for focal planes with a different number of bands, nodes can easily be configured to any number of colors.

The combined output of the worker nodes goes to toggle buffers memory-mapped onto VME buses. The DSI interface

boards, also developed by Rockwell, take the latest FPA buffers and send them to the seeker. The number of DSI boards needed is a function of the number of separate bands and the data rates required per band. The TC2000 supports five VME buses for every eight nodes and provides a very flexible architecture for different FPA configurations and rates.

Conclusion

The SISG design and testing were accomplished within 6 months of the BBN TC2000 delivery. The system is now on line to support the SBI program. Due to the generic and modular design employed, the system can be easily reconfigured to support other programs. For instance, the seeker proposed to be used in the Army's antisatellite (ASAT) system can be driven with the SISG. With data base changes and minor code modifications, the scene generator can produce the single-color, visible-band FPA image required. Only modifications of the seeker side of the DSI boards is required to provide the proper electrical handshaking necessary for the different brassboard hardware. The scalable software and hardware design can even support multiple seekers concurrently. In addition, because of the MIMD architecture of the TC2000 processors, all simulation host processing tasks (equations of motion, propulsion, mass properties, etc.) have been allocated to

run on nodes in the TC2000 not used for scene generation. This allows the entire simulation to be performed in one integrated system, increasing flexibility, efficiency, and productivity and allowing unlimited growth.

In summary, the successful completion of the SISG project resulted from several important development steps. First the optimal allocation of functions to off-line and on-line tasks minimized real-time computational/processor requirements. Next, the use of a representative benchmark routine (that accurately reflects the worst-case computations) to evaluate potential computer platforms resulted in the selection of a computer system that successfully performed to expectations. Finally, the use of generic and modular software techniques resulted in a final system that achieved all goals and was very easy to reconfigure and modify to meet changing program requirements.

Two Step Method for Parameter Estimation of Nonlinear Systems

Wang Zicai Wang Jinhua Zhao Changan
Harbin Institute of Technology, China

Abstract

For nonlinear systems with process and measurement noises which are white noises with unknown mean values and variances, a two step method—adaptive Kalman filter plus least square method for identification of such a system is presented in this paper. Using the method suggested, it is possible to estimate the parameters of the system and the characteristic parameters of the noises. The example shows that the algorithm is convergence at high speed.

Introduction

For a linear system with known parameters and noise characteristics, the Kalman filter can be used to obtain the optimal estimate of states. For a nonlinear system, the generalized Kalman filter is often used to estimate states, but the condition to use it is that the statistic characteristic of noises must know priori. In real systems, the statistic characteristics of noises is often unknown.

If the states estimate of system are based on incorrect statistic characteristics, the large estimate error will be produced even then divergence [1]. Even though the statistic characteristics of noises in the nonlinear system are known, the estimate still exists error due to linearization error in generalized Kalman filtering process. The concept of invented noise is introduced in [2]. The linearization error is combined into system noise to constitute the invented noise and use it to compensate linearization error of model. The mean value and variance of the invented noise is estimated gradually in the

filtering process and adaptive Kalman filter for an estimator of nonlinear system with time-variant invented noise is formed. Using this algorithm, the estimator may be divergence in some case. In order to overcome this disadvantage, An improved time-variant noise estimator is suggested here by using filter error to form a weighting factor which is used to correct the noise estimator.

In order to estimate the unknown parameters of the system, the states must be measured, and then estimate system parameters using some classical identification methods. In practice, the states of system is often unmeasurable, so it is impossible to use some classical methods to estimate the parameters directly. The two step method i.e., adaptive Kalman filter plus least square method for identification is presented in this paper.

Improved invented time-variant noise estimator

1. Invented noises

Consider a discrete nonlinear system given by

$$\begin{aligned} x(k+1) &= f[x(k), \theta(k)] + w(k) \\ y(k+1) &= h[x(k+1)] + v(k) \end{aligned} \quad (1)$$

$k = 0, 1, 2, \dots$

where $x(k+1)$ is the n -dimension state vector at $k+1$, $y(k+1)$ is the m -dimension measured vector at $k+1$, $\theta(k)$ is the s -dimension parameter vector at k , $w(k)$ and $v(k)$ are independent each other Gaussian white noises with mean value and covariance

$$\begin{aligned} E[w(k)] &= q, \quad \text{cov}[w(k), w(j)] = Q\delta_{kj} \\ E[v(k)] &= r, \quad \text{cov}[v(k), v(j)] = R\delta_{kj} \end{aligned} \quad (2)$$

where $E[\cdot]$ is the mean operation sign, cov is the

covariance sign, δ_{ij} is Kronecker function.

If q, Q, r and R are unknown and we have obtain the state estimate value $\hat{x}(k)$.

Expanding $f[x(k), \theta(k)]$ in the neighbourhood of $x(k)$ into Taylor series

$$f[x(k), \theta(k)] = f[\hat{x}(k), \theta(k)] + \frac{\partial f}{\partial \hat{x}(k)} \times [x(k) - \hat{x}(k)] + H.O.T \quad (3)$$

where H.O.T is all high order terms of Taylor expansion. Now equation (1) can be written as

$$x(k+1) = \frac{\partial f}{\partial \hat{x}(k)} x(k) + u(k) + \xi(k) \quad (4)$$

where

$$u(k) = f[\hat{x}(k), \theta(k)] - \frac{\partial f}{\partial \hat{x}(k)} \hat{x}(k) \\ \xi(k) = w(k) + H.O.T$$

Clearly, even though the mean value of $w(k)$ is zero, the mean value of $\xi(k)$ is nonzero.

Let $\Delta x(k) = x(k) - \hat{x}(k)$, then H.O.T = $O(\Delta x(k))$. If $\Delta x(k)$ is small, then $O[\Delta x(k)]$ is also small. In this case the noise characteristic of $w(k)$ is not strong dependent of H.O.T, and only mean value and variance of $w(k)$ change a little. Using extended noise compensation technique, the $\xi(k)$ can be approximated as a white noise with unknown time-variant statistic characteristic which is called extended state noise.

Let $E[\xi(k)] = q(k)$, $\text{cov}[\xi(k), \xi(j)] = Q(k)\delta_{kj}$.

As the same way, expanding h in the neighbourhood of $\hat{x}(k+1/k)$, we have the linearized measurement equation

$$y(k+1) = \frac{\partial h}{\partial \hat{x}(k+1/k)} x(k+1) + z(k+1) + \eta(k+1) \quad (5)$$

where

$$z(k+1) = h[\hat{x}(k+1/k) - \frac{\partial h}{\partial \hat{x}(k+1/k)} \hat{x}(k+1/k) \\ \eta(k+1) = v(k+1) + H.O.T$$

The $\eta(k+1)$ can also be approximated as a white noise with unknown time-variant statistic characteristic, which is called extended measurement noise. Let $E\eta(k+1) = r(k+1)$, $\text{cov}[\eta(j+1), \eta(j+1)] = R(k)\delta_{kj}$.

2. Algorithm

For system (1), an algorithm of adaptive Kalman filter for time-variant noise estimator was proposed in [1].

The filter equation are

$$\hat{x}(k+1) = \hat{x}(k+1/k) + K(k+1)e(k+1) \\ \hat{x}(k+1/k) = f[\hat{x}(k), \theta(k)] + \hat{q}(k) \\ e(k+1) = y(k+1) - h[\hat{x}(k+1/k)] - \hat{p}(k) \\ K(k+1) = P(k+1/k) \\ \times \left(\frac{\partial h}{\partial \hat{x}(k+1/k)} \right)^T \left[\left(\frac{\partial h}{\partial \hat{x}(k+1/k)} \right) P(k+1/k) \right. \\ \left. + \left(\frac{\partial h}{\partial \hat{x}(k+1/k)} \right)^T + \hat{R}(k) \right]^{-1}$$

$$P(k+1/k) = \left(\frac{\partial f}{\partial \hat{x}(k)} \right) P(k) \left(\frac{\partial f}{\partial \hat{x}(k)} \right)^T + \hat{Q}(k) \\ P(k+1) = [I - K(k+1) \left(\frac{\partial h}{\partial \hat{x}(k+1/k)} \right)] \\ \times P(k+1/k) \quad (6)$$

The time-variant noise estimator is

$$\hat{q}(k+1) = [1 - d(k)] \hat{q}(k) + d(k) \hat{x}(k+1) - f[\hat{x}(k), \theta(k)] \\ \hat{Q}(k+1) = [1 - d(k)] \hat{Q}(k) + d(k) [K(k+1) \\ e(k+1)e^T(k+1)K^T(k+1) + P(k+1) \\ - \left(\frac{\partial f}{\partial \hat{x}(k)} \right) P(k) \left(\frac{\partial f}{\partial \hat{x}(k)} \right)^T] \\ \hat{p}(k+1) = [1 - d(k)] \hat{p}(k) + d(k) [y(k+1) - h[\hat{x}(k+1/k)]] \\ \hat{R}(k+1) = [1 - d(k)] \hat{R}(k) + d(k) [e(k+1) \\ \times e^T(k+1) - \left(\frac{\partial h}{\partial \hat{x}(k+1/k)} \right) P(k+1/k) \\ \times \left(\frac{\partial h}{\partial \hat{x}(k+1/k)} \right)^T] \quad (7)$$

where

$$d(k) = (1-b) / (1-b^{k+1}), \quad 0 < b < 1$$

In the algorithm, the estimate of noise statistic characteristics is obtained from the estimated and filter error before this instant. When the filter error is large, due to the weighting factor $d(k)$ is related to the step number and is unrelated to the filter error, the norm of estimate value in next step is larger than in this step, ..., the estimate can be divergence. In order to overcome the divergence, a correct factor $\alpha(k)$, which is determined by the norm of filter error, is introduced. Now, we have

$$\hat{q}(k+1) = \hat{q}(k) + \alpha_1(k)d(k)K(k)e(k) \\ = \hat{q}(k) + \alpha_1(k)d(k)\{\hat{x}(k+1) - f[\hat{x}(k), \theta(k)] - \hat{q}(k)\}$$

or

$$\hat{q}(k+1) = [1 - \alpha_1(k)d(k)] \hat{q}(k) + \alpha_1(k)d(k) \\ \times \{\hat{x}(k+1) - f[\hat{x}(k), \theta(k)]\}$$

We can treat other formulas above as the same way, the improved noise estimator is

$$\hat{q}(k+1) = [1 - \alpha_1(k)d(k)] \hat{q}(k) + \alpha_1(k)d(k) \\ \times \{\hat{x}(k+1) - f[\hat{x}(k), \theta(k)]\} \\ \hat{Q}(k+1) = [1 - \alpha_2(k)d(k)] \hat{Q}(k) + \alpha_2(k)d(k) \\ \times [K(k+1)e(k+1)e^T(k+1)K^T(k+1) \\ + P(k+1) - \left(\frac{\partial f}{\partial \hat{x}(k)} \right) P(k) \left(\frac{\partial f}{\partial \hat{x}(k)} \right)^T] \\ \hat{p}(k+1) = [1 - \alpha_3(k)d(k)] \hat{p}(k) + \alpha_3(k)d(k) \\ \times \{y(k+1) - h[\hat{x}(k+1/k)]\} \\ \hat{R}(k+1) = [1 - \alpha_4(k)d(k)] \hat{R}(k) + \alpha_4(k)d(k) \\ \times [e(k+1)e^T(k+1) - \left(\frac{\partial h}{\partial \hat{x}(k+1/k)} \right) \\ \times P(k+1/k) \left(\frac{\partial h}{\partial \hat{x}(k+1/k)} \right)^T]$$

$$\begin{aligned}
& \times P(k+1/k) \left(\frac{\partial h}{\partial \hat{x}(k+1/k)} \right)^T \Big] \\
\alpha'_1(k) &= \|q(k+1)\| / \|\hat{x}(k+1) - f[\hat{x}(k), \theta]\| \\
\alpha'_2(k) &= \|\hat{Q}(k)\| / \|K(k+1)e(k+1)e^T(k+1) \\
& \quad K^T(k+1) + P(k+1) - \left(\frac{\partial f}{\partial \hat{x}(k)} \right) P(k) \left(\frac{\partial f}{\partial \hat{x}(k)} \right)^T \Big] \\
\alpha'_3(k) &= \|\hat{r}(k)\| / \|y(k+1) - h[\hat{x}(k+1/k)]\| \\
\alpha'_4(k) &= \|\hat{R}(k)\| / \|e(k+1)e^T(k+1) \\
& \quad - \left(\frac{\partial h}{\partial \hat{x}(k+1/k)} \right) P(k+1/k) \left(\frac{\partial h}{\partial \hat{x}(k+1/k)} \right)^T \Big] \\
\alpha_1(k) &= \begin{cases} \alpha'_1(k), & \alpha'_1(k) < 1 \\ 1, & \alpha'_1(k) \geq 1 \end{cases} \\
\alpha_2(k) &= \begin{cases} \alpha'_2(k), & \alpha'_2(k) < 1 \\ 1, & \alpha'_2(k) \geq 1 \end{cases} \\
\alpha_3(k) &= \begin{cases} \alpha'_3(k), & \alpha'_3(k) < 1 \\ 1, & \alpha'_3(k) \geq 1 \end{cases} \\
\alpha_4(k) &= \begin{cases} \alpha'_4(k), & \alpha'_4(k) < 1 \\ 1, & \alpha'_4(k) \geq 1 \end{cases}
\end{aligned}$$

Parameter estimates of system using least square criterion

In the system (1), let $\theta^*(k)$ represent the true value of $\theta(k)$ and belong to a set $[a, b]$ with s -dimension. Assume $\theta^*(k)$ is known. then in (6), (8), (9), $\theta(k)$ can be substituted by $\theta^*(k)$, and using this substitution, the good results for states filter are obtained. If $\theta^*(k)$ is unknown, then we can use $\theta(k)$ to substitute $\theta(k)$ in (6), (8) and (9). Let choose the estimation criterion as

$$J_k[\theta(k)] = \sum \|e(k+1)\|^2 \quad (10)$$

where

$$e(k+1) = y(k+1) - h[x(k+1/k)] - r(k)$$

The parameter vector $\theta(k)$ can be got by minimizing the criterion (10) using the least square algorithm, we have

$$y(k+1) - r(k+1) = h[x(k+1/k)] + n(k) \quad (11)$$

where $n(k)$ is the white noise.

Using Kalman filter method the estimate value of system parameter can be obtained. Using the estimate value to estimate the states again. Repeat this procedure, we have the new recursive algorithm.

i. Given a set of initial values of system parameters, the system is filtered by using the generalized Kalman filter with extended noise estimator.

ii. Based on the filter results, fulfil the least square estimate of system parameters by using algorithm (11), and get a new set of system parameters.

iii. Using the new system parameters to filter the sys-

tem again, the procedure is processed until the satisfactory results are obtained.

Example

Consider an aircraft system, in which the aerodynamic parameters are depended on flight condition and some attitude variables, the relationship between them can be expressed as a polynomial function. Now, the identification of parameters of the aircraft system can be transformed to the estimate problem of coefficients of polynomial. Let the dynamic model of aircraft is of the form

$$\begin{aligned}
x_1(k+1) &= [23.5 - 2.54 \times 10^{-4} a_1 x_1^2(k) - 2.386 \\
& \quad \times 10^{-7} x_1^3(k) - 1.96 \times 10^{-9} a_2 x_1^4(k) \\
& \quad - (292.04 - 10.025k) \sin x_2(k)] / (29.8 \\
& \quad - 1.023k) + w_1(k)
\end{aligned}$$

$$\begin{aligned}
x_2(k+1) &= [1.6403 + 2.54 \times 10^{-4} a_3 x_1^2(k) + 7.06 \\
& \quad \times 10^{-7} a_4 x_1^3(k) - 9.604 \times 10^{-11} x_1^4(k) - \\
& \quad (292.04 - 10.025k) \cos x_2(k)] / \\
& \quad (29.8 - 10.23k) + w_2(k)
\end{aligned}$$

The measurement equation is

$$y(k) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} v_1(k) \\ v_2(k) \end{bmatrix}$$

where x_1 is the velocity of the aircraft; x_2 is the pitch angle of the aircraft; a_1, a_2, a_3, a_4 are unknown parameters; $w_1(k), w_2(k), v_1(k), v_2(k)$ are white noise sequences.

Using the algorithm, the estimate results of a_1, a_2, a_3, a_4 are shown in Fig.1.2.3.4.

Conclusion

In model identification, the estimated accuracy will reduce even divergence due to uncertainty of noise characteristic and structure error of model. Introducing invented-noise concept incorporated with estimated noise characteristic, it is possible to increase accuracy of the estimated parameters and improve the convergence speed.

It is well known that the parameter estimation is related to the state estimation. Considering the characteristics of approximation to true values for both parameter and state estimation, the two-step method for parameter estimation i.e. repeating filter-estimation processes, it is able to increase convergence speed and improve accuracy of estimation. The simulation results show that the method presented is correct and effective.

Reference

- (1) Anderson, B.D.O. & J.B.Moore "Optimal filtering" Preaticce-Hall, Inc.1979.
- (2) Deng Zili and Wang Jianguo "Adaptive generalized Kalman filter of nonlinear systems" Journal of Automation, China, Sept,1987.

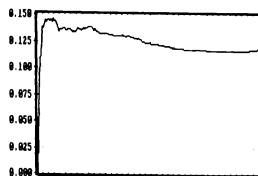


Fig.4. parameter a_4



Fig.1. parameter a_1

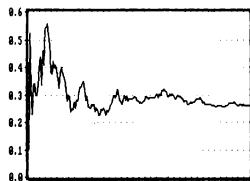


Fig.2. parameter a_2

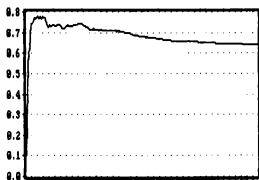


Fig.3. parameter a_3

AN IMPROVED METHOD OF INTEGRATING THE EQUATIONS OF MOTION

Mark S. Fineberg
McDonnell Aircraft Company
A Division of
McDonnell Douglas Corporation

Abstract

A key element in the digital simulation of continuous systems is the process of numerical integration. The necessity of providing more accurate answers for more complex systems at less cost requires the most efficient possible integration methods. A new method is presented which has proven superior to all others suggested.

This paper first presents two principles which pertain to the accuracy of numerical integration. These principles are presented and a brief justification of their validity is given.

Double integration is required to implement the second principle. A family of double integration formulae is derived. These formulae have errors proportional to the sample interval raised to various powers. Family members are shown with powers ranging from two to six. This power determines the order of each formula. There are several formulae of each order, one to be compatible with each of the several single integration formulae.

These formulae were first tested for their performance as isolated elements by using truncation errors and by plotting the frequency response. They were then used to construct second order systems and the transient response was used as the primary criterion for evaluating various methods and different order formulae.

Two apparent problems are discussed along with their solutions.

The new method was used on a full F/A-18 simulation which is typical of the most detailed simulations we run. The results matched those run on simplified systems.

Based on these tests a set of formulae is recommended which is roughly a factor of two better than the older ones that we had been using.

Introduction

To simulate a continuous system we usually start with a set of differential equations. The derivatives, which are defined in the statement of the differential equations, are evaluated continuously in the real system. If a digital computer is to be used a sampled system must be built. That requires the derivatives to be evaluated once (or maybe several times) for each sample interval. The process of simulation is really the process of integrating the deriv-

atives. Therefore numerical integration is the cornerstone of digital simulation of continuous systems. The method chosen for integration plays a vital role in determining the accuracy, stability, and cost of the simulation.

It is helpful to start by discussing the process of numerical integration, stressing those points which will provide insight for the following analysis. Integration may be considered the measuring of area under a curve. But in a sampled data system there is no curve. There are only points at the sample instants and there is no area under a set of points. A guess at some reasonable curve through the points must be made before an area can be calculated. One way of looking at the various approaches to integration is to break the process into four conceptual steps. The first step is to gather input values. The second step is to create a continuous function or curve which fits the data points and is reasonable across the interval of interest. Next, calculate the area under the freshly created curve. Finally, a value which represents the area is presented to the rest of the system. Of these processes, the first, third, and fourth are trivial and usually error free. That means that all the significant error is generated in building a continuous function to approximate the input!

There are two different situations that must be discussed. One is when there are input data points at both the beginning and end of the interval to be considered. The other is when data exists at the beginning and prior to the beginning of the interval but there is no data at the end. In the first case a curve may be drawn connecting the end points. For a higher order approximation, points prior to the interval may be used to estimate the curvature in the middle. In the other case there is no data at the far end of the interval so that data points before the interval must do double duty. In addition to being used to estimate the curvature in the middle of the interval they must also serve to estimate the value of the function at the end. Figure 1 sketches the situation. It is obvious that not knowing the value of the function at the far end of an interval dramatically reduces the accuracy of any estimate of the area beneath it.

Naming of these two cases is a bit of a problem. Using the terms "closed" and "open" describes a mathematical characteristic of the interval. (For instance see Hildebrand¹.) The end points of a "closed" interval are contained in the interval but the end points are not considered to be part of an "open" interval. The term "predictor" is nicely descriptive of the

open interval but the complementary name almost always used in conjunction with "predictor" is "corrector". This is because a very popular multipass method is called the "predictor-corrector". In that multipass method the first pass is an open form which is the predictor pass. That is followed by one or more closed passes in which the derivative is repeatedly evaluated, each evaluation corresponding to exactly the same time. These are the corrector passes. This is continued until some criterion is reached and then the routine goes on to the next interval. The utilization of the closed form for homing in on a better solution by using many derivative evaluations for the same time is very well described by the term "corrector". But its use is confusing when describing the one pass open/closed algorithm advocated here. The terms "explicit" and "implicit" do describe a functional relationship. In closed forms the derivative is a function of the integrator output corresponding to the same time as the derivative. When the value of the derivative is substituted into an integration formula the current value appears on both sides of the equal sign. That makes the equation implicit. On the other hand with open (predictive) forms the difference equation becomes an explicit function of old values of the function together with values of the input. Hence it is termed explicit. None of these is totally satisfactory. The "open" - "closed" terminology is most often used in the following discussions.

The First Principle

The first principle states that the number of open forms (predictors) in each computational path should be minimized. Consider motion along a single axis. Assume we know the value of velocity and position at some time N . The acceleration at time N can then be calculated from the differential equation describing the problem. The crucial process is to integrate both the velocity and position to be valid for time $N+1$. If the position is taken first, an open form integration can be applied to give an estimate of position at time $N+1$ using values of velocity and position at time N . (For higher order approximations older values will also be used.)

The same process may be applied to velocity. Integrating acceleration at time N yields a value for velocity at $N+1$. Note that two open (predictive) integrators were used. But what if the order of processing is reversed and velocity is integrated first? The position integration must use a velocity from time N for its input to be consistent with the open form. This is true even though a new and better value of velocity has been calculated and is available. But if a closed form with velocity at $N+1$ for its input is used for the position integration there is only one predictor in the computational path. Application of our first principle means that the method using only one predictor is preferred.

There are a variety of ways to verify the validity of applying this principle. Evaluating the transient response, looking at the frequency

response, and comparing the truncation error will all add to our understanding. Consider second order formulae, those which have an error proportional to the square of the time between samples. The simplest error criterion to evaluate is the truncation error since it is expressed as a single metric. It is well known that a trapezoidal predictor (open) has an error proportional to $5/12$ and the Tustin integrator (closed) has an error proportional to $-1/12$. (For instance see Howe².) That means we expect to reduce errors by a factor of 5 if the closed form can be used. The equivalent calculation for third and fourth order integrators show ratios of 9 to 1 and 13.2 to 1 improvement.

Figure 2 is a plot of amplitude and phase errors as a function of frequency for the Tustin and trapezoidal predictor integrators. It shows that the Tustin (closed) form has no phase error until half the sample frequency. But the trapezoidal predictor (open) has considerable phase error. Comparing the magnitude errors shows that the closed integration has a much more benign nature. The gain of the Tustin is less than one which means it tends to be more stable whereas the predictor has a gain larger than one and a phase lag, both of which contribute to artificial instability.

The transient response of an integrator by itself is not particularly interesting. The more interesting case of a loop closed around various types of integrators is discussed later. But one important characteristic is shown clearly. Figure 3 shows the open and closed integrator response to a step. Two frame times after the step is applied neither method shows any error for this simple case. But there is a great difference in the period just after the step. Since the predictor does not have the latest information the effects of a change in the input are at least one frame late. That is an important performance characteristic. Replacing a predictor with a closed form always decreases the transport delay by one frame!

This discussion has shown an intuitive rationale for using closed integration instead of open whenever it is feasible. Several tests have shown that it ought to outperform the more traditional approaches. Discussion of more relevant performance criteria and more complex systems is deferred until some additional points have been covered.

Using closed integrators along with open ones is neither a new or an original idea. Since there are major advantages to these methods why are they not used universally? The reason is probably the extra complexity of having several types of integrator to worry about. Unless it is realized that there are advantages in using closed forms there is no incentive to accept the slight extra complexity.

The case we are most interested in is the equations of motion used to solve Newton's second law. Since it relates the acceleration to position there is a double integration implied. That means that half the integrations are prime candidates for the closed form.

The Second Principle

The second principle states that the number of samplers in each path should be minimized. It seems obvious that introducing superfluous samplers is a silly thing to do. But extra samplers often sneak in unintentionally. Most of our intuition has been acquired in continuous linear systems where it makes no difference if several blocks are combined into one or a single block is split up. However in a sampled data system whenever a block is split into two it is the equivalent of inserting an extra sampler. Likewise, when two blocks are made into one it can be the same as removing a sampler. Our intuition may be saved by recognizing that in a continuous system each of the blocks communicates with its neighbors at every instant whereas in sampled systems neighbors share information only at the sampling instants.

If the blocks concerned are an open integrator followed by a closed one we see that there is a sampler between them. If we replace the open/closed combination with a double integrator we are removing a sampler. Block diagrams showing each combination are presented in Figure 4.

Considering the four steps for numerical integration discussed above, it is evident that when two blocks are combined, the logical equivalent to the steps of gathering input and fitting a curve are not required for the operations corresponding to the second block. Since most of the error is introduced when fitting the curve, eliminating its necessity eliminates the primary source of error.

Derivation of Double Integration Formulae

Applying the second principle requires open form double integrators. Where do we get them? Unfortunately my search of the literature helped very little so I was forced to derive most of them. The one I did find is:

$$P_{N+1} = 2 \cdot P_N - P_{N-1} + h^2 \cdot A_N,$$

where h is the time interval between samples. This formula may be shown to have an error proportional to h^2 , which means that it is a member of the second order family (error $\rightarrow h^2$) which includes the Tustin integrator and trapezoidal predictor.

The most likely way to run across this double integration formula is via an argument similar to the following. Compare Euler integration formulae for both open and closed intervals. It may be seen that the open form and closed form are algebraically identical. Also, when used as an open form the error approximates a half frame delay. When used as a closed form the error looks like a half frame lead. This prompts the question: Why don't we put an open and closed together so that their errors compensate? Indeed we can and when analyzed the result turns out to be exactly the one shown above.

This very simple predicting double integrator has some remarkable properties. There is no

phase error at all! When it is used to simulate the response of a undamped continuous system the computation will also be perfectly undamped. This is not a new discovery but it is not often used in a general context.

This, and most other integration formulae, may be derived in many ways. In this simple case the easiest way is to use pole mapping. The continuous double integrator has two poles at $S = 0$. These poles map to $Z = 1$ in the Z domain. The Z transform therefore has two poles at $Z = 1$ and no zeros. That makes it the simplest possible form with two poles. When casting the resulting Z -transform into the equivalent difference equation we see immediately that it is the same as shown above.

It is also very interesting to note that the method called "modified Euler" by Howe³, which is based on alternate half interval sampling, can also be shown to be the same as this formula for the double integration. All the benefits he cites apply equally here.

There is also a way this formula may be derived entirely in the time domain. Assume that the acceleration during the entire interval is constant and equal to the value sampled at the beginning of the interval, i.e. the Euler assumption. Integrating the constant twice gives a parabolic formula in terms of old velocities and positions. Assuming that the velocity is computed using the trapezoidal predictor formula, a velocity term may be substituted for one of the position terms. The result is a new formula for position which is a function of old positions and accelerations only. It is indeed the same as shown above.

The method of deriving these formulae that I find most useful is to start with a generalized Z transform where all the coefficients are unknown and represented by symbols. Then set the Z transform equal to one over S^2 . Replace each occurrence of Z with its power series in S . The result is an equation in powers of S . By letting terms in each power of S become separate equations we can get as many equations as there are coefficients. Solving these equations gives values for the Z transform coefficients. The first few terms of the Taylor expansion for the error are all zero because the method used to derive the coefficients forces those terms to zero. Each additional unknown coefficient in the Z transform gives one extra degree of freedom which gives one more equation and thus one more zero term in the series for the error. That way we can construct as high an order approximation as we like. This process for deriving Z transform coefficients was discussed in more detail earlier (Fineberg 89), although in a different context. Table 1 shows formulae with errors that range from being proportional to h^2 through formulae with errors proportional to h^6 . There are also various formulae for use in conjunction with different single integration methods. The single integrator for use with each double integration formula is shown in table 2 with each formula labeled by an "O" or "C" (for Open or Closed) followed by a single digit denoting the order of the formula.

Choosing Appropriate Formulae

Armed with our two principles and formulae necessary to implement them we can compare different methods of integration. Figure 4 illustrates the basic configurations of interest. They will be called open/open, open/closed, and open/double. Note that when properly described the extra one frame delay of the open/open configuration becomes obvious. The open/closed form is the result of applying the first principle. The open/double formula is obtained by using the first and second principles.

To illustrate how the various formulae are related the second order are shown below.

The only second order velocity formula is:

$$V_{N+1} = V_N + \frac{h}{2}(3A_N - A_{N-1})$$

For open/open integration the same formula may be applied again, this time to position.

$$P_{N+1} = P_N + \frac{h}{2}(3V_N - V_{N-1})$$

For open/closed the Tustin formula is used for position.

$$P_{N+1} = P_N + \frac{h}{2}(V_{N+1} + V_N)$$

The open/double formula for position is:

$$P_{N+1} = 2P_N + P_{N-1} + h^2 A_N$$

The non-redundant form of the open/double is:

$$P_{N+1} = P_N + hV_N + \frac{h^2}{2}(2A_N - A_{N-1})$$

To compare these methods we will take the easiest criterion first. The truncation error for the open/open case is calculated to be proportional to five sixths of the square of both the step size and the natural frequency of the system being tested. For the open/closed situation the same calculations show an error coefficient of one third, which is two and a half times better.* If a double integrator is used the coefficient is one twelfth, or four times better than the open/closed combination and ten times better than two open forms cascaded.

The frequency response errors of second order integration methods are presented in Figure 5. As can be seen, using the open/open is far the worst. The open/closed response shows considerable improvement. The open/double is significantly better. The same trends are also seen for higher order integrators.

Generally, if the input is smooth, higher order formulae should have less error. However in actual simulations there are three practical constraints which can cause problems when using higher orders. The first is stability. The extra terms in the difference equation which

allow better accuracy do so by creating extra poles in the Z transform. These poles will move toward the unit circle as the natural frequency of the simulated system goes higher. As these unwanted poles move towards the circle the system becomes unstable. Higher order forms create more poles and make them start closer to the circle of divergence. That instability often limits how high an order can be used.

The second reason to avoid high orders is that they require taking the difference of large, nearly equal numbers. That accentuates round off errors. To be confident of correct results it would be wise to stay well away from that region of error.

The third problem depends on the input seen by the integrators. An input which has abrupt changes can be described as having a large high frequency content or it may be said to have large coefficients for high order terms in its Taylor series expansion. Conversely smooth inputs correspond to small high frequency content and negligible magnitude of high order terms in the Taylor series. For smooth inputs high orders can be extremely accurate since the approximation used for the estimation of the input is itself accurate. But when the input is abrupt, high order methods tend to have a tortured response and degraded accuracy. Empirical methods will be used to determine the best order for a specific situation.

Comparison

What has been discussed so far can help us to understand the differences between the various methods but comparing them in a more realistic setting is necessary before we can intelligently choose formulae to use for real problems. The test environment used here is a second order system with a nominal damping ratio of 0.3. A sample interval of one second was used to make scaling to real situations easy and to keep the computation as simple as possible. To represent problems that we often run, the natural frequency used was about one tenth of the sampling frequency. A traditional sample rate in our shop was 20 Hz and an airframe was in the neighborhood of 2 Hz. When scaled to a 1 Hz sample rate the natural frequency becomes 0.6 radians.

The shape of the test input can also affect the results. As discussed above, smooth inputs favor high orders and abrupt favors low orders. A doublet is used in testing real airplanes so it seemed ideal for our purposes. It is reasonably abrupt but is continuous. Our test input was a triangular signal going positive then negative and back to zero. To make it look more realistic a first order lag with a time constant of 0.3 seconds filtered the triangle.

The plots for these comparisons show the doublet input, the ideal continuous response, and three approximations, one the for open/open configuration, one for open/closed, and one for the double integrator. These correspond to the traditional approach, applying the first

* What looks like an inconsistency in the above numbers may be observed. Earlier it was pointed out that replacing a trapezoidal predictor with a Tustin is expected to cut the error by a factor of five whereas in this case a factor of two and a half was reported. The reason is that the error of the open form is $\pm 5/12$ while the closed error is $-1/12$. Therefore the errors have opposite signs so when used together they partially cancel.

principle, and applying both principles. The first comparison (figure 6) is for the second order case. It shows that the open/open response is tending to go unstable and not very accurate. When the natural frequency is reduced (the equivalent of increasing the sample rate) it appears to be more reasonable but not comparable to the alternatives.

In the third order response (figure 7) the open/open computation shows very little resemblance to the desired output. The open/closed and open/double are both improved compared to the second order case and again the double is better.

The fourth order (figure 8) has errors which (for open/closed and open/double) are similar to the third order. The open/double still looks a little more accurate than the open/closed. In the fifth order (figure 9) the open/double has gone unstable.

In all the tests discussed so far the order of both the velocity and position integration were the same. But it is quite feasible to use different orders for the two integrations. Since abruptness is the enemy of higher orders, and the position integrator sees an input that is less abrupt because it has been filtered by the velocity integrator, an argument could be made that the position integrator should be of a higher order. Five different combinations where the two integrations are of different order were tried and compared to the ones discussed earlier. Each of the new combinations was tried for the open/closed and the open/double configurations.

Figure 10 shows the second/third order case. (The orders of the velocity and position integrators will be indicated by giving the order of each separated by a "/". Thus second/third means the velocity integration is second order while the position integrator is third order). Figure 11 is for the third/fourth, figure 12 for the second/fourth, figure 13 for the third/fifth, and figure 14 for the fourth/fifth order cases. The open/double is always seen to be better than the open/closed.

The order 3/3 and 3/4 are quite similar and are the best tested. Based on close examination of the very complex model (the F/A-18 described later) the 3/4 was judged to be slightly superior. These results confirm what would be expected from the preceding qualitative discussion. They show that applying the principles does indeed improve accuracy. They also show that, for problems with parameters similar to ours, the third/fourth order formulae are appropriate.

Stability is also an important characteristic to examine. It can be the factor which limits the order to be used which, in turn, limits the accuracy. A series of tests were run to explore the stability margin for the various methods. Each configuration was set up and the natural frequency was adjusted until it just entered into a constant amplitude oscillation. Table 3 shows at what natural frequency the each configuration goes unstable. The relative stability of each combination of orders may then be compared.

Stability is seen to be degraded more strongly as the order of the velocity integrator is increased than when the order of the position integrator is raised. (The open/open is an exception but it is not of much interest.) Table 3 also shows that the open/closed form is more stable than the open/double. Therefore, if our primary concern is to make the simulation as stable as possible, even if accuracy is degraded, the second order open/closed is the best choice.

Thus far all the methods considered have been single pass and therefore applicable to real time simulation, the area of most interest to us. However these principles are equally valid for almost any situation. One of the most respected multipass integration methods is known by the name Runge-Kutta. To illustrate the performance of the new methods they were compared to Runge-Kutta. The same test used before was applied to the Runge-Kutta of orders two and four and compared with the open/double of order 3/4. The results are presented in figure 15. It might seem surprising at first to see that the Runge-Kutta has points calculated at larger time intervals than the open/double. The appendix explains why this is necessary. It can be seen that the Runge-Kuttas hardly compare with the desired output at all while the open/double of order 3/4 is quite reasonable. The same stability test used above may be repeated for the Runge-Kutta formulae. The second order has a stability parameter of 0.82 and the fourth has a parameter of 0.73. That shows that they are slightly more stable than the open/double but not nearly as stable as the open/closed.

Application

In practical application of these formulae there is one booby trap which must be avoided. Looking at the double integration formulae, we see that there are two old values of position which correspond to the two constants of integration required for double integration. There is also a velocity formula which has one old value needed for its constant of integration. But two points of position history imply a value for velocity, or vice versa. Therefore one of the values is redundant. When implemented the two redundant integrators start out equal to one another but eventually they accumulate small differences which can grow without bound. The solution is to use alternate forms which have no redundancy. To derive such an alternate formula we recognize that the formula for velocity relates values of V 's to A 's. So, with a little algebra, we can substitute a V for the third P in the position formula. The new formula to be used for double integration has one old position and one old velocity in addition to acceleration terms. This process depends on which velocity formula is used. Table 1 has several entries for each order, one corresponding to each of the single integration methods.

Rotational Equations

The formulation is quite practical and does exhibit the advantages predicted. However, the rotational equations, because of the geometry

involved, don't have explicit double integrations. The version of the equations of motion we use starts with the angular accelerations. One integration is performed to get the angular rates. Instead of Euler angle integration we use quaternion integration. Four quaternion rates (Edot1, Edot2, Edot3, and Edot4) are calculated from equations of the form:

$$\text{Edot}_i = -(E_2 p + E_3 q + E_4 r) / 2,$$

where E_1 through E_4 are the quaternions, p , q , and r are the angular rates and Edot is the derivative of E . The Edots are then integrated to get the E 's. The Euler angles theta, phi, and psi, are calculated from E 's. There is no obvious double integration involved.

The nature of the problem may be seen by looking at the right hand side of the Edot equation. It shows Edot to be made up of terms containing the E 's multiplied by an angular velocity. Since new E 's can not be available until the Edots have been integrated and the Edots are functions of the E 's we can not use closed integrators. New values of the angular rates are available but they can not be used because it is incorrect to multiply new rates by old quaternions.

However, the Edot equations may be differentiated again. When the differentiation is carried out the resulting equations for quaternion acceleration are of the form:

$$\begin{aligned} \text{Edoubledot}_1 = & -(E_2 p\dot{p} + \text{Edot}_2 p + E_3 q\dot{q} \\ & + \text{Edot}_3 q + E_4 r\dot{r} + \text{Edot}_4 r) / 2. \end{aligned}$$

If open double integration is used, the terms on the right hand side of the Edoubledot equation are needed at time N so there is no problem of using inconsistent or late values. Even though E 's and p , q , and r do not have a direct derivative-integral relationship, they may serve as the input for the E double integrations. That means that the non-redundant double integrator can easily be used. The same reformulation may be applied to the open/closed formulae. (That would make sense for implementing the second order open/closed form when extra stability is desired.)

Full Scale Tests

To test to these formulae, they were incorporated in one of our most detailed simulations, the handling qualities F/A-18 model. Actually some of the tests described above were done first on the detailed airframe. The results from the F/A-18 were quite close to those from the simpler model. The same formulae had the least error. The third/fourth had a very slight advantage over the third/third. Overall the best choice is the open/double 3/4 which is nearly the best in every test. The suggested set of formulae is:

$$V_{N+1} = V_N + \frac{h}{12}(23A_N - 16A_{N-1} + 5A_{N-2})$$

$$P_{N+1} = P_N + hV_N + \frac{h^2}{6}(7A_N - 7A_{N-1} + 3A_{N-2}).$$

Conclusion

Two principles were discussed from an intuitive point of view. A new integration scheme based on these principles was presented and supported by showing intuitive arguments and testing isolated components. Then a series of tests were run on a more realistic but still simplified system. Finally a full handling qualities F/A-18 was run. The very detailed model displayed the same improvements seen in the simpler tests.

The primary advantage of this method is its accuracy. Often that accuracy may be traded off for cost savings by using a larger time interval so that the same accuracy can be obtained with less calculation. It is more stable than most methods currently in use. However, where the utmost stability is more important than accuracy, the second order open/closed formulae should be used. The new method also has a single frame time delay while the most often used methods cause a delay of two frames. We have found that if we take some error criterion and determine at what frequency that error occurs and then try it with the new method, the same magnitude error usually occurs about an octave higher. Also, if we use lower and lower sample rates until the system goes unstable, we find it usually goes unstable about an octave lower. Thus we can say that it is about a factor of two improvement over what we have been using.

Even though the new method was devised for real time simulation, comparison with Runge-Kutta methods shows that it should be equally valuable wherever numerical integration is used.

Appendix

In figure 15 the Runge-Kutta have points plotted at larger time intervals than the open/double. The Runge-Kutta-2 is double the interval and the Runge-Kutta-4 is four times the interval. There are several ways to explain why this is the only reasonable comparison to be made.

In all but the most trivial problems the bulk of the computing effort is spent evaluating the derivative. Therefore the computing cost for each step when using the Runge-Kutta methods is two or four times as much as that for a single pass method. By adjusting the sample rates we are comparing methods which require the similar computing resources.

To look at the situation another way, assume for the moment that accuracy per time step is taken to be a relevant criterion. One obvious approach for the integration would be to try some well known multipass method such as Runge-Kutta. It is theoretically possible to derive Runge-Kutta formulae of any order but the calculations get very difficult as the order is increased. We might instead derive a new hypothetical multipass method. This new method would be based on some reasonable single pass method which might be called its prototype. The new method would divide the interval up into k equal subintervals. Each of the intermediate

calculations would use the same formulae as the prototype single pass method, but for a smaller step size. A weighted average of the intermediate answers would be taken (just like Runge-Kutta). In our new method the weighting function would be particularly simple. The weight of the last value would be one and all the other weights would be zero. But this new method is identical to applying the prototype while using a step size k times smaller and then ignoring $k-1$ of each k points. That means that for any order multipass method we can build another method which uses the same number of derivative evaluations. We have shown that for orders 2 and 4 the hypothetical method is superior to the Runge-Kuttas. There is no reason to believe that higher order Runge-Kuttas will reverse the trend. Even if there is a better multipass method we could modify our hypothetical method to use a smaller interval and be of higher order. This would produce arbitrarily small error and great stability without violating the assumed definition of single step. Since we could make a very large number of derivative evaluations to make the error arbitrarily small

there must be something wrong with the underlying assumptions. That implies that to be realistic we must use accuracy per derivative evaluation to judge the accuracy of integration schemes.

References

1. Hildebrand, F .B., *Introduction to Numerical Analysis*, McGraw-Hill, 1956
2. Howe, R. M., "Transfer Function and Characteristic Root Errors for Fixed-Step Intergration Algorithms", Transactions of the Society for Computer Simulation, 2 (4), pp 320, 1988
3. Howe, R. M., "The Role of Modified Euler Integration in Real-Time Simulation", Aerospace Simulation, 1988
4. Fineberg, M. S., "A New Method for Optimally Fitting Z Transform to Laplace Transforms", AIAA Flight Simulation Technologies Conference, 1989

Figure 1a
CLOSED INTERVAL

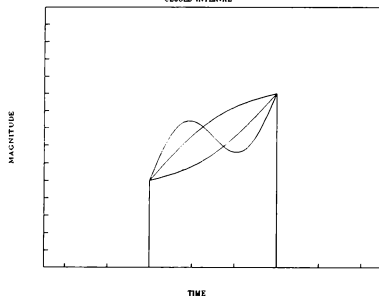


Figure 1b
OPEN INTERVAL

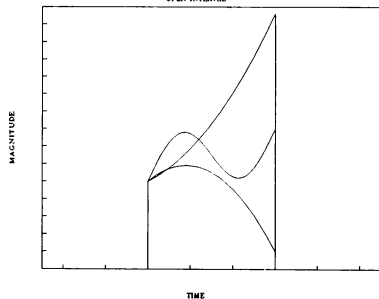


Figure 2a
MAGNITUDE ERROR

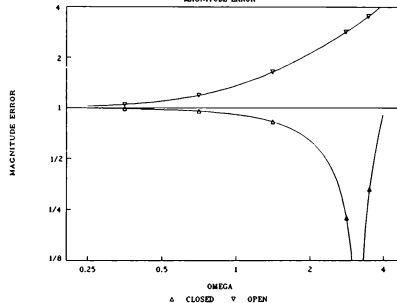
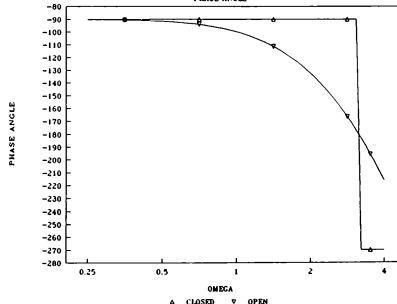


Figure 2b
PHASE ANGLE



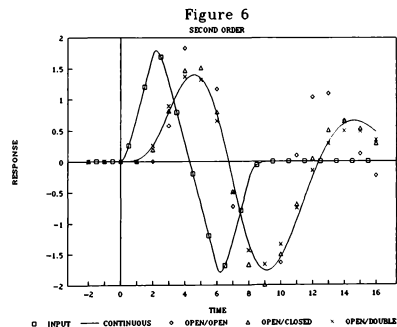
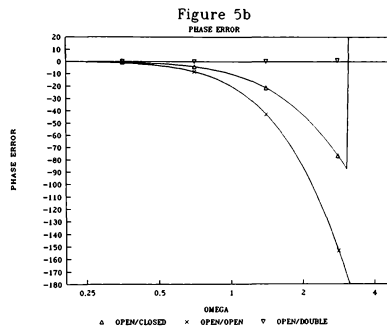
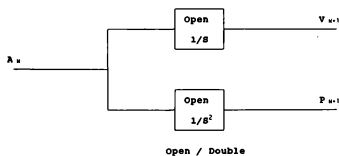
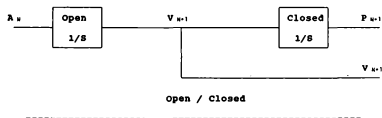
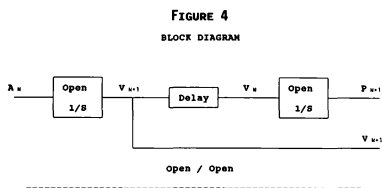
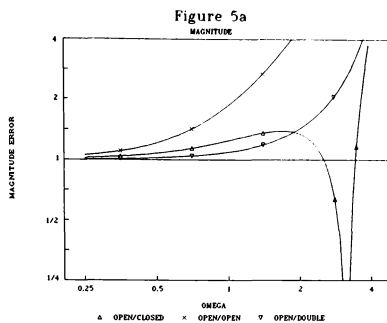
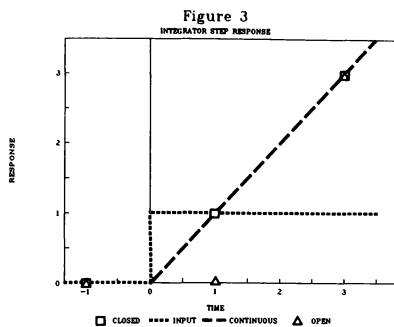


Figure 7

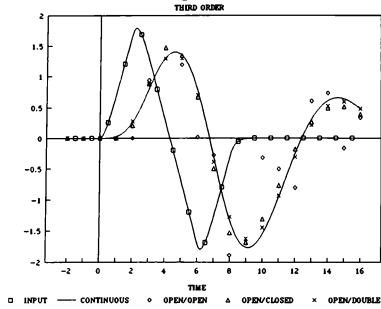


Figure 10

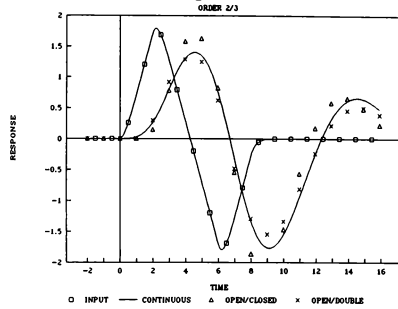


Figure 8

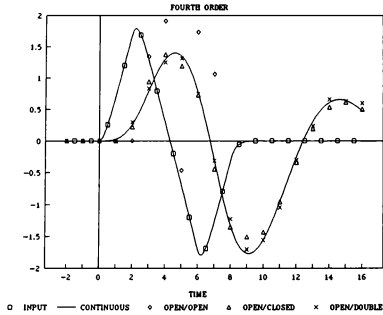


Figure 11

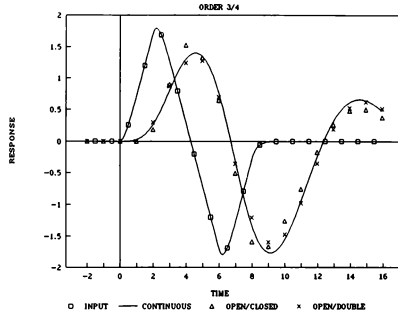


Figure 9

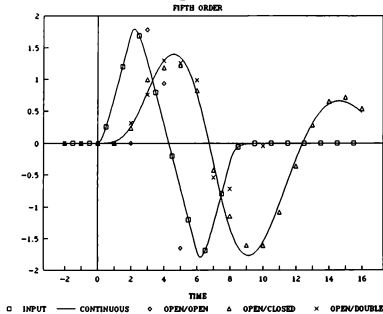


Figure 12

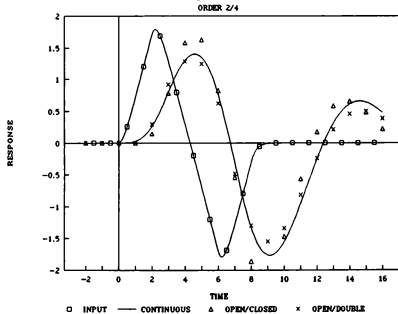


Figure 13

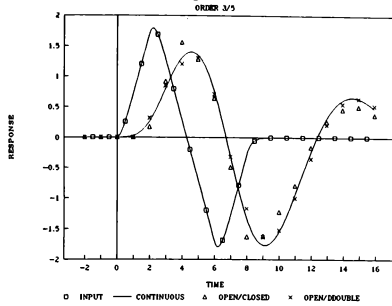


Figure 14

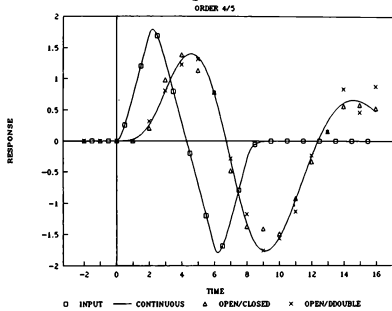
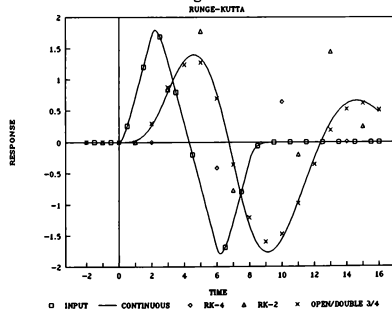


Figure 15



DOUBLE INTEGRATORS

Second Order

$$P_{N-1} = 2P_N - P_{N-1} + h^2 A_N + 0 \cdot A_{N-1}$$

with 02,
 $P_{N-1} = P_N + h^2 V_N + \frac{h^2}{2} (2A_N - A_{N-1})$

with 03,
 $P_{N-1} = P_N + h^2 V_N + \frac{h^2}{12} (12A_N - 11A_{N-1} + 5A_{N-2})$

Third Order

$$P_{N-1} = 2P_N - P_{N-1} + \frac{h^2}{12} (13A_N - 2A_{N-1} + A_{N-2})$$

with 02,
 $P_{N-1} = P_N + hV_N + \frac{h^2}{12} (13A_N - 7A_{N-1})$

with 03,
 $P_{N-1} = P_N + hV_N + \frac{h^2}{12} (13A_N - 12A_{N-1} + 5A_{N-2})$

with 04,
 $P_{N-1} = P_N + hV_N + \frac{h^2}{12} (26A_N - 33A_{N-1} + 28A_{N-2} - 9A_{N-3})$

Fourth Order

$$P_{N-1} = 2P_N - P_{N-1} + \frac{h^2}{12} (14A_N - 5A_{N-1} + 4A_{N-2} - A_{N-3})$$

with 02,
 $P_{N-1} = P_N + hV_N + \frac{h^2}{12} (14A_N - 9A_{N-1} + A_{N-2})$

with 03,
 $P_{N-1} = P_N + hV_N + \frac{h^2}{6} (7A_N - 7A_{N-1} + 3A_{N-2})$

with 04,
 $P_{N-1} = P_N + hV_N + \frac{h^2}{24} (28A_N - 37A_{N-1} + 30A_{N-2} - 9A_{N-3})$

Fifth Order

$$P_{N-1} = 2P_N - P_{N-1} + \frac{h^2}{240} (299A_N - 176A_{N-1} + 194A_{N-2} - 96A_{N-3} + 19A_{N-4})$$

with 02,
 $P_{N-1} = P_N + hV_N + \frac{h^2}{240} (299A_N - 237A_{N-1} + 77A_{N-2} - 19A_{N-3})$

with 03,
 $P_{N-1} = P_N + hV_N + \frac{h^2}{240} (299A_N - 337A_{N-1} + 177A_{N-2} - 19A_{N-3})$

with 04,
 $P_{N-1} = P_N + hV_N + \frac{h^2}{240} (299A_N - 247A_{N-1} + 357A_{N-2} - 109A_{N-3})$

with 05,
 $P_{N-1} = P_N + hV_N + \frac{h^2}{720} (897A_N - 1532A_{N-1} + 1824A_{N-2} - 1080A_{N-3} + 251A_{N-4})$

Sixth Order

$$P_{N-1} = 2P_N - P_{N-1} + \frac{h^2}{240} (317A_N - 266A_{N-1} + 374A_{N-2} - 276A_{N-3} + 109A_{N-4} - 18A_{N-5})$$

with 02,
 $P_{N-1} = P_N + hV_N + \frac{h^2}{240} (317A_N - 309A_{N-1} + 185A_{N-2} - 91A_{N-3} + 18A_{N-4})$

with 03,
 $P_{N-1} = P_N + hV_N + \frac{h^2}{240} (317A_N - 409A_{N-1} + 285A_{N-2} - 91A_{N-3} + 18A_{N-4})$

with 04,
 $P_{N-1} = P_N + hV_N + \frac{h^2}{240} (317A_N - 499A_{N-1} + 465A_{N-2} - 181A_{N-3} + 18A_{N-4})$

with 05,
 $P_{N-1} = P_N + hV_N + \frac{h^2}{720} (951A_N - 1748A_{N-1} + 2148A_{N-2} - 1296A_{N-3} + 305A_{N-4})$

with 06,
 $P_{N-1} = P_N + hV_N + \frac{h^2}{1440} (1902A_N - 3971A_{N-1} + 6196A_{N-2} - 5442A_{N-3} + 2510A_{N-4} - 475A_{N-5})$

TABLE 1

SINGLE INTEGRATORS

- 02: Second Order Open -

$$V_{n+1} = V_n + \frac{h}{2} (3A_n + A_{n-1})$$
- C2: Second Order Closed -

$$V_n = V_{n-1} + \frac{h}{2} (A_n + A_{n-1})$$
- 03: Third Order Open -

$$V_{n+1} = V_n + \frac{h}{12} (23A_n + 16A_{n-1} + 5A_{n-2})$$
- C3: Third Order Closed -

$$V_n = V_{n-1} + \frac{h}{12} (5A_n + 8A_{n-1} + A_{n-2})$$
- 04: Fourth Order Open -

$$V_{n+1} = V_n + \frac{h}{24} (55A_n + 59A_{n-1} + 37A_{n-2} + 9A_{n-3})$$
- 05: Fourth Order Closed -

$$V_n = V_{n-1} + \frac{h}{24} (9A_n + 19A_{n-1} + 5A_{n-2} + A_{n-3})$$
- 05: Fifth Order Open -

$$V_{n+1} = V_n + \frac{h}{720} (1901A_n + 2774A_{n-1} + 2616A_{n-2} + 1274A_{n-3} + 251A_{n-4})$$
- C5: Fifth Order Closed -

$$V_n = V_{n-1} + \frac{h}{720} (251A_n + 646A_{n-1} + 264A_{n-2} + 106A_{n-3} + 19A_{n-4})$$
- 06: Sixth Order Open -

$$V_{n+1} = V_n + \frac{h}{1440} (4277A_n + 7923A_{n-1} + 9982A_{n-2} + 7298A_{n-3} + 2877A_{n-4} + 475A_{n-5})$$
- C6: Sixth Order Closed -

$$V_n = V_{n-1} + \frac{h}{1440} (493A_n + 1337A_{n-1} + 618A_{n-2} + 302A_{n-3} + 83A_{n-4} + 9A_{n-5})$$

TABLE 2

RELATIVE STABILITY

ORDER OF VELOCITY INTEGRATION

ORDER OF POSITION INTEGRATION		2 nd	3 rd	4 th		2 nd	3 rd	4 th	5 th		2 nd	3 rd	4 th
	2 nd	1.13	.77	.47		1.67	.91	.51	.29		.831	.85	.75
	3 rd	1.05	.74	.47		1.30	1.32	.6	.3		.75	.64	.52
	4 th	.94	.69	.45		1.15	1.08	.31	.31		.62	.48	.37
	5 th	.9	.62	.62		1.04	.98	.4	.4		.47	.35	.26
		OPEN / DOUBLE				OPEN / CLOSED					OPEN / OPEN		

TABLE 3

THE USE OF FUNCTION GENERATION IN THE REAL-TIME SIMULATION OF STIFF SYSTEMS

R. M. Howe* and K. C. Lin **
University of Michigan
Ann Arbor, MI 48109-2140

Abstract

A new approach, the function generation method, is suggested in this paper to simulate the stiff dynamic systems. The equations of the fast subsystem are integrated off-line over a time interval which will be used as the step size for the on-line integration of the slow subsystem. This off-line integration uses a sufficiently small step size to insure both accurate and stable solutions and is repeated for a matrix of initial conditions and inputs. These results are then stored in multivariable function tables. In the on-line real-time simulation, table lookup and linear interpolation are then used to determine each new fast subsystem state based on the old state and inputs.

A time-shift scheme is also introduced. In this scheme, the frame times for the fast subsystem are shifted half an integration step with respect to the integer frame times for the slow subsystem. This, in turn, means that the output data sequences of the fast and slow subsystems are staggered by one-half of an integration step with respect to each other. This scheme provides substantial improvement in accuracy of the function generation method.

Three practical examples are used to demonstrate how the function generation method is implemented. From these examples it is shown that the function generation method has speed and accuracy advantages over conventional integration of all state equations with a common step size. It is also shown that the time-shift scheme can improve the accuracy of the function generation method.

1. Introduction

One of the most difficult real-time simulation problems is the simulation of stiff dynamic systems. These systems are governed by equations which, when linearized, have eigenvalues that are widely separated. Implicit methods that are commonly used

in nonreal-time simulation of such systems require iterations within each integration step and are not generally suitable for real-time implementation¹. On the other hand, explicit integration methods normally used in real-time simulation require very small step sizes and excessive computing power in simulating stiff systems.

In 1985 Palusinski² suggested a multiple frame-rate integration technique. Howe³ has applied multiple frame-rate integration to real-time simulation of two-speed systems. Until now, this has been the only non-standard integration method for dealing with real-time simulation of stiff systems.

In 1977 Gilbert and Howe³ suggested that some intrinsic mathematical functions such as sines, cosines, reciprocals, etc., can be computed more rapidly by table lookup and interpolation rather than the traditional method of series evaluation. Recently, Kabamba⁴ has used pre-stored shape functions to mechanize sampled data control systems. Because of the increasing availability of low-cost high-speed memory, there is a trend toward more and more use of function generation in simulation, especially in real-time computation.

In 1988 Howe⁴ analyzed the state transition method for simulation of constant-coefficient linear dynamic system. In this method the state transition matrix of the system is pre-calculated and stored. In the on-line real-time simulation, the state-variable vector at the current time step is multiplied by the state transition matrix to obtain the state-variable vector at the next time step. This method, although efficient for the time-critical real-time simulation, is only applicable to the linear systems.

2. Basic Assumptions

The systems considered in this paper are represented by the two state vector equations

$$\dot{x} = \mathcal{F}(x, y, u) \quad (1)$$

$$\dot{y} = \mathcal{G}(y, x, v) \quad (2)$$

where \mathcal{F} and \mathcal{G} are nonlinear derivative functions, x and y are state variables, and u and v are input func-

*Professor of Aerospace Engineering
Associate Fellow, AIAA

**Research Assistant in Aerospace Engineering
Currently, Asst. Professor, University of Central Florida

tions. Assume that the variable z in general changes much more rapidly than the variable y . z is called the fast variable and the z -equation the fast equation or fast subsystem. Similarly, the variable y is called the slow variable and the y -equation the slow equation or slow subsystem. The system is a stiff system. The integration step size must be chosen based on the stability requirement of the fast equation. This may require a step which is too small for real-time simulation using a given digital computer.

There are two basic assumptions which will be made in this paper:

1. The most important factor in real-time simulation is computational speed². The accuracy requirement for real-time simulation is usually moderate. Computational speed can not be sacrificed for improvements in accuracy or saving in computer memory if the resulting simulation does not keep up with real time.
2. Off-line computation time is more or less unlimited. In practice, of course, it is impossible to have unlimited computer resources for off-line preparation of the on-line simulation. However, the time spent in off-line preparation of a real-time simulation can often be much longer than the time required for the real-time simulation itself.

3. Function Generation Method

The function generation method is motivated by the state transition matrix method. The procedure is the following:

- Decouple the fast and slow subsystems; treat the slow state variable as a forcing function input for the fast equation, and the fast state variable as a forcing function input for the slow equation.
- Integrate the two subsystems separately; the fast subsystem is integrated off-line before the actual simulation is executed; the slow subsystem is integrated on-line with a step size h based on the dynamic accuracy and stability requirement of the slow subsystem.
- Update the two state variables each integration step; the slow variable is updated using a real time integration algorithm; the fast variable is updated using function generation.

The fast subsystem is integrated off-line over a total time interval h with various discrete values of x as initial conditions, and various discrete values of y and u as forcing function inputs. For this off-line solution the integration step size can be made as small as necessary to obtain the required dynamic accuracy and

stability. From the off-line solution the state variable x at the end of the time interval h is tabulated as a function of x at the beginning of the interval, and y and u . For example, if y and u are only defined at the beginning of the interval, the variable x at the next time step is

$$x_{n+1} = f^x(x_n, y_n, u_n) \quad (3)$$

There are more accurate methods for representing y and u over the time interval h which will not be discussed here⁷.

The function f^x can be viewed as a nonlinear state transition matrix for the nonlinear fast subsystem represented by Eq. (1). If the step size for the off-line integration of the fast subsystem is so small that the dynamic errors can be neglected, the function generation method becomes exactly analogous to the state transition matrix method for linear systems, where the only error is that caused by the method used to approximate the continuous input over the integration step^{4,7}.

4. Time Shift Scheme

In the time shift scheme the integer frame times for the fast variable x are shifted half an integration step with respect to the integer frame times for the slow variable y . This in turn means that the x and y data sequences are now staggered by one-half integration step with respect to each other. In the off-line integration of the fast subsystem the error caused by representing y as a constant over the interval h is then of order h^2 , as opposed to order h for the non-time-shift case⁷.

5. Nonlinear Flight Control

Fig. 1 is the block diagram of the nonlinear flight control system used as the first example. The system consists of a nonlinear controller subsystem representing an actuator, a proportional controller with gain constant k_a , proportional plus rate feedback with rate-constant C_q , and a pure inertia plant representing the aircraft.

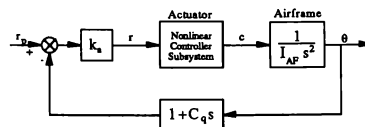


Figure 1. Nonlinear flight control system.

The block diagram of the nonlinear controller subsystem is shown in Fig. 2. The nonlinear con-

troller consists an effort-limited proportional controller with gain constant k and a lead-lag controller with rate-constant C_d and lag-constant τ . The nonlinear effort-limited proportional controller is actually a piecewise linear controller with a linear region for $y_2 \leq y \leq y_1$ and two saturated regions for $y > y_1$ and $y < y_2$, respectively. The plant is a pure inertia plant.

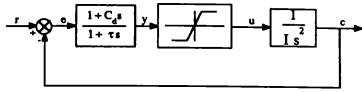


Figure 2. Nonlinear controller subsystem.

The differential equation of the airframe is simply

$$I_{AF} \ddot{\theta} = c \quad (4)$$

Here c is the output from the nonlinear controller subsystem. The differential equation of the nonlinear controller subsystem is

$$I \ddot{c} = u(y) \quad (5)$$

where $u(y)$ is an effort-limited function defined as

$$u = \begin{cases} u_1, & \text{if } y > y_1 \\ k \cdot y, & \text{if } y_2 \leq y \leq y_1 \\ u_2, & \text{if } y < y_2 \end{cases} \quad (6)$$

Here u_1, u_2 are constants, and

$$y = x + C_d \dot{x} \quad (7)$$

where x is a state variable governed by the differential equation

$$\tau \dot{x} + x = e \quad (8)$$

and

$$e = r - c \quad (9)$$

The input r to the nonlinear controller subsystem is the output from the flight control system and is given by

$$r = k_a(r_p - \theta - C_q \dot{\theta}) \quad (10)$$

Table 1 shows the parameters used for this example. For these parameters the nonlinear controller subsystem of Fig. 2, by itself, has a dominant characteristic root pair corresponding to an undamped natural frequency of 10 rad/sec and a damping ratio of 0.2 when operating within the linear region of the controller. If we neglect the dynamics of the nonlinear controller subsystem, then the flight control system of Fig. 1, for the parameters in Table 1, has an undamped natural frequency of 1 rad/sec and a damping ratio of 0.5. Hence this is a stiff system with

Table 1. Parameter Values for the Flight Control Example.

parameter	value	parameter	value
C_d	0.05	C_q	1
τ	0.01	k_a	1
k	100	I_{AF}	1
I	1	e_2	-1
e_1	1	u_2	-100
u_1	100		

the airframe as the slow subsystem and the nonlinear controller as the fast subsystem. The function generation method is applied to this example. When integrating the fast subsystem nonlinear controller off-line, we assume that the slow input variable remains constant over the integration step, i.e., $r = r_n$. In this case the states c_{n+1} , \dot{c}_{n+1} , and x_{n+1} are functions of four variables, c_n , \dot{c}_n , x_n , and r_n . Then, each step in the on-line simulation the AB-2 algorithm is used to integrate the $\dot{\theta}$ equation, and the four-variable function generation method is used to update the c equation. In the off-line determination of the entries for the four-variable function table, 17 values of each of the four inputs, c_n , \dot{c}_n , x_n , and r_n , were used in constructing the table. Fig. 3 shows the simulation result for a step input $r_p(t) = 2$ using the function generation method with an integration step size $h = 0.25$ second. The simulation is stable, whereas for this step size any conventional real-time integration algorithm will be unstable. Also, the accuracy of the time history of pitch angle output θ is reasonable, despite the fact that there are substantial errors in the time history of the actuator output c . Fig. 4 shows the result when the function generation method with time shifting is applied to this example. In this case the fast variable c is updated at half-integer frame times and the slow variable θ is integrated at integer frame times. It is evident that the time shift scheme does provide substantially improved accuracy in the time histories of both θ and c .

6. Airframe/Landing-gear

The second example is a simplified model for an airframe and landing gear. Fig. 5 shows the model. The mass of the airplane is represented by m_1 . It is significantly greater than m_2 , which is the mass of the landing gear. The two masses are interconnected with a linear spring and damper to model the suspension system. Only one dimension of the motion, vertical displacement, is considered. y_1 and y_2 are chosen as the coordinates which describe the

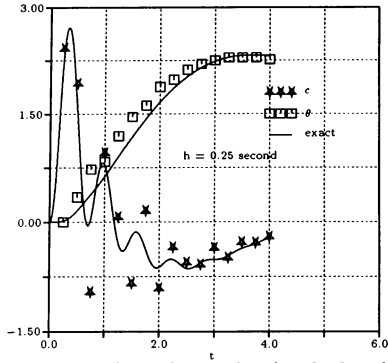


Figure 3. Simulation result using the function generation method.

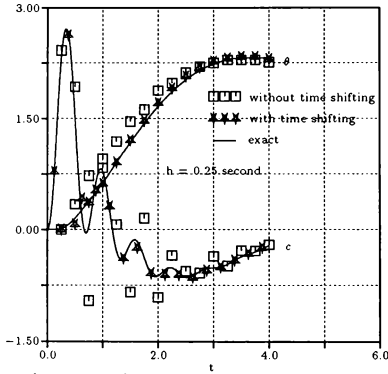


Figure 4. Simulation results using the function generation method with and without time shifting.

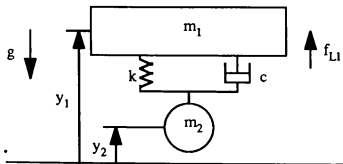


Figure 5. Airframe/landing-gear system.

positions of masses m_1 and m_2 , respectively. Displacement is measured from the ground level, i.e., the runway. The system is subjected to a constant aerodynamic lift force f_{L1} acting on the airplane. The entire system is moving downward in flight until it hits the hard ground and bounces up. The system will eventually stop after several bounces.

It is evident that this is a system with a unilateral constraint. Real-time numerical solution of a differential equation system with a unilateral constraint presents major difficulties, since the number of degrees of freedom are reduced by one (here from two to one) when the constraint is encountered. To avoid this problem we assume that the landing gear is not rigid, but consists of a tire represented with both a spring and damper, as shown in Fig. 6. The differential equations of this new model are given by

$$m_1 \ddot{y}_1 + c(\dot{y}_1 - \dot{y}_2) + k(y_1 - y_2 - l_0) = f_{L1} - m_1 g \quad (11)$$

$$m_2 \ddot{y}_2 + c(\dot{y}_2 - \dot{y}_1) + k(y_2 - y_1 + l_0) + c_e \dot{y}_2 + k_e y_2 = -m_2 g \quad (12)$$

Here the tire spring constant k_e is given by

$$\begin{cases} k_e = 0, & \text{if } y_2 \geq 0 \\ k_e = \text{constant} > 0, & \text{if } y_2 < 0 \end{cases} \quad (13)$$

and the damping constant c_e by

$$\begin{cases} c_e = 0, & \text{if } y_2 \geq 0 \\ c_e = \text{constant} > 0, & \text{if } y_2 < 0 \end{cases} \quad (14)$$

In order to make this model behave approximately like the original constrained system, the spring k_e must be very stiff. This will in turn cause stability problems in the numerical solution, since it will add a pair of very large eigenvalues to the simulation when the ground is encountered ($y_2 < 0$). The function generation method can handle this type of problem efficiently.

In the system represented in Fig. 6 the landing-gear subsystem is stiff compared with the airframe.

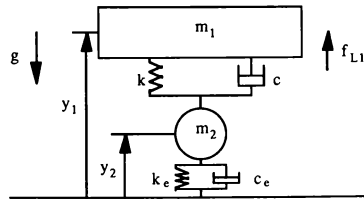


Figure 6. Modified model of the airframe/landing-gear system.

From Eq. (12) the landing-gear subsystem equation is rewritten as

$$m_2 \ddot{y}_2 + (c + c_e) \dot{y}_2 + (k + k_e) y_2 = c \dot{y}_1 + k y_1 - m_2 g - k l_0 \quad (15)$$

Here the airframe displacement y_1 and velocity \dot{y}_1 are considered to be part of the forcing function. When the function generation method is applied, the state variables \dot{y}_2 and y_2 at step $n+1$ depend on the variables \dot{y}_2 , y_2 , \dot{y}_1 , and y_1 at the current and/or previous steps. The landing-gear subsystem is now integrated off-line with the assumption that y_1 and \dot{y}_1 remain constant over the time interval $[t_n, t_{n+1}]$. The state variables y_2 and \dot{y}_2 are then functions of three variables only, and can be tabulated as

$$\dot{y}_{2,n+1} = f^{\dot{y}_2}(y_{2,n}, \dot{y}_{2,n}, C_n) \quad (16)$$

$$y_{2,n+1} = f^{y_2}(y_{2,n}, \dot{y}_{2,n}, C_n) \quad (17)$$

where

$$C_n = c \dot{y}_{1,n} + k y_{1,n} \quad (18)$$

Then, each integration step in the on-line simulation a conventional real time algorithm is used to integrate the y_1 and \dot{y}_1 equations represented by Eq. (11), and the three variable function generation represented by Eqs. (16) and (17) is used to update the y_2 and \dot{y}_2 .

Instead of assuming that y_1 remains constant over the time interval $[t_n, t_{n+1}]$, we can assume that \dot{y}_1 remains constant and y_1 changes linearly over the interval $[t_n, t_{n+1}]$. Then

$$y_1(t) = y_{1,n} + \dot{y}_{1,n} t, \quad 0 \leq t \leq h \quad (19)$$

In this case the state variables y_2 and \dot{y}_2 are functions of four variables. From off-line simulation they can be tabulated as

$$\dot{y}_{2,n+1} = f^{\dot{y}_2}(y_{2,n}, \dot{y}_{2,n}, y_{1,n}, \dot{y}_{1,n}) \quad (20)$$

$$y_{2,n+1} = f^{y_2}(y_{2,n}, \dot{y}_{2,n}, y_{1,n}, \dot{y}_{1,n}) \quad (21)$$

Table 2 shows the parameters used for this example. When the airframe subsystem is considered separately by neglecting the mass of the landing gear, the resulting eigenvalues $\lambda_{y1,1,2} = -0.0391 \pm j1.9612$. This corresponds to an undamped natural frequency $\omega_{n,1} = 1.9616$ rad/sec and a damping ratio $\zeta_1 = 0.0204$. Considered separately, the fast subsystem landing-gear has eigenvalues $\lambda_{y2,1,2} = -25 \pm j98.87$. These correspond to an undamped natural frequency $\omega_{n,2} = 101.98$ rad/sec and damping ratio $\zeta_2 = 0.245$.

Fig. 7 shows the results of the function generation method when three and four-variable functions are used with a step size of 0.05 second. Comparison with the exact solution shows, as expected, that the four-variable function leads to better accuracy.

Table 2. Parameter Values for the Airframe /Landing-gear Example.

parameter	value	parameter	value
m_1	10,000	m_2	100
k	40,000	k_e	1,000,000
c	4,000	c_e	1,000
f_{L1}	320,000	l_0	4.97

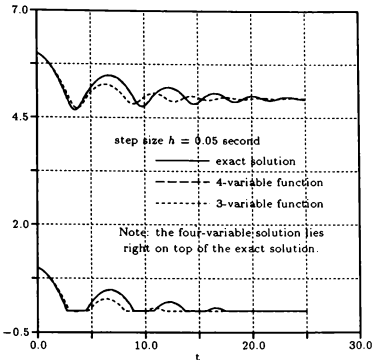


Figure 7. 3- and 4-variable function generation.

This is because the airframe motion over the interval $[t_n, t_{n+1}]$ is represented as a linear time function in the four-variable case, as opposed to being represented as a constant over the interval $[t_n, t_{n+1}]$ in the three-variable case. Note that conventional real-time integration algorithms will be unstable for $h = 0.05$ second.

The accuracy of the function generation method using three variables, however, can be improved considerably by the time shift scheme. This is evident in the results shown in Fig. 8. Recall in this case that the fast subsystem is updated at half-integer frame times and the slow subsystem is updated at integer frame times.

7. Tracked Vehicle

The third example is a tracked vehicle running on a hard but not rigid terrain. The vehicle consists of a chassis and seven pairs of road wheels, as illustrated in the side view of Fig. 9. Each road wheel has the same radius, which is assumed to be large compared with the track pitch. Track pitch is defined as the length of the rigid track segments that are connected together to form the overall track. Since only the

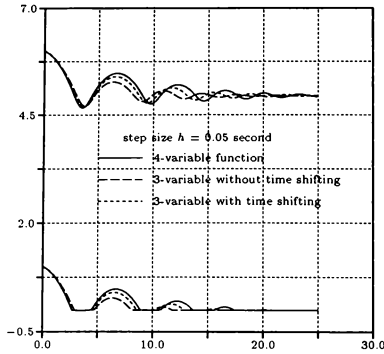


Figure 8. 3-variable function generation with time shifting.

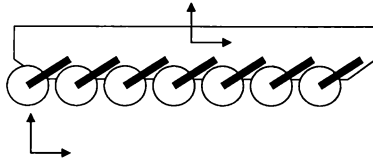


Figure 9. Pitch plane model of tracked vehicle.

two-dimensional pitch-plane motion of the vehicle is considered, knowing the motion of seven of the road wheels will by symmetry give the motion of the other seven.

The suspension of the vehicle is composed of four-ten sets of torsional bars and shock absorbers, which connect the road arms of the fourteen road wheels to the chassis. Fig. 10 shows the road arm and road wheel system, where the torsional bar and the shock absorber are modeled as a linear spring-damper system with restoring torques given by

$$T_{tb} = k_{tb}(\psi - \psi_n) \quad (22)$$

$$T_{sa} = c_{sa}\dot{\psi} \quad (23)$$

where

- T_{tb} = torque of torsional bar,
- T_{sa} = torque of shock absorber,
- k_{tb} = spring constant of torsional bar,
- c_{sa} = damping coefficient of shock absorber,
- ψ = road arm angle relative to chassis,
- ψ_n = road arm angle for which $T_{tb} = 0$,

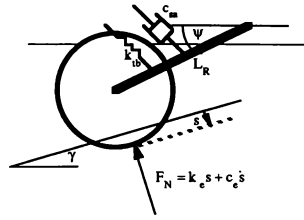


Figure 10. The road arm and road wheel system of the vehicle.

$\dot{\psi}$ = road arm ang. vel. relative to chassis.

When represented in symmetric, pitch-plane motion, the system consists of one chassis and seven road arms, that is, eight interconnected rigid bodies. The chassis has three degrees of freedom, two translational (horizontal and vertical) and one rotational (pitch). Each road arm has one degree of freedom, i.e., rotation relative to the chassis. In total, there are ten degrees of freedom. Two coordinate systems are chosen: (x_s, y_s, z_s) fixed in space, and a body-axis system (x_h, y_h, z_h) fixed in the chassis with origin at the center of mass (c.m.) of the chassis. The orientation of the body axes (x_h, y_h, z_h) coincides with the principle axes of the chassis. Ten generalized coordinates are used to describe the motion of the system:

- (x_c, y_c) = chassis c.m. pos. in space axes,
- ψ_h = chassis pitch ang. w.r.t. space axes,
- ψ_1 = road arm #1 ang. relative to chassis,
- \vdots
- ψ_7 = road arm #7 ang. relative to chassis.

The corresponding generalized speeds are

- u_h = x_h comp. of the chassis c.m. vel.,
- v_h = y_h comp. of the chassis c.m. vel.,
- r_h = $\dot{\psi}_h$ = pitch rate of the chassis,
- $\dot{\psi}_1$ = ang. rate of road arm #1 relative to chassis,
- \vdots
- $\dot{\psi}_7$ = ang. rate of road arm #7 relative to chassis.

After appropriate simplifications, the equations of motion are

$$\dot{u}_h = v_h r_h + F_x / m_{tot} \quad (24)$$

$$\dot{v}_h = -u_h r_h + F_y / m_{tot} \quad (25)$$

$$\dot{r}_h = T_z/I_{tot} \quad (26)$$

$$\dot{\psi}_i = T_i/I_i, \quad i = 1, \dots, 7 \quad (27)$$

Here m_{tot} is the total mass of the system and I_{tot} is the total moment of inertia of the system about the z_h (pitch) axis. Thus

$$m_{tot} = m_h + \sum_{i=1}^7 m_i$$

$$I_{tot} = I_h + \sum_{i=1}^7 I_i$$

where m_h is the mass of the chassis, I_h is the moment of inertia of the chassis about z_h (pitch) axis, m_i is the mass of the i th road arm, I_i is the moment of inertia of the i th road arm about its pivot point, and

F_x = x_h component of total applied forces,
 F_y = y_h component of total applied forces,
 T_z = total applied torque about z_h (pitch) axis,
 T_i = torque of i th road arm about pivot point.

The applied forces of the system include gravity and forces from the terrain. The applied torques about the chassis c.m. include the torques of the torsional bars and shock absorbers, and the torques of the applied forces. The applied torque of the i th road arm about its pivot point includes the torsional bar torque T_{tb} , the shock absorber torque T_{sa} , and the terrain normal-force torque, which is given by

$$T_{ne} = F_N \cdot L_R \cos(\psi_{rh} - \gamma) \quad (28)$$

Here F_N is the normal reaction force of the terrain, as shown in Fig. 10. All other forces and torques are neglected. The terrain profile is assumed to be piecewise linear, and the normal reaction force F_N from the terrain is assumed to be proportional to the maximum penetration of the road wheel, as shown in Fig. 10. Thus F_N is given by

$$F_N = k_e s + c_e \dot{s} \quad (29)$$

where s is the maximum penetration of the road wheel, \dot{s} is the rate of penetration, and k_e and c_e represent effective spring and damping constants that depend on the properties of the terrain.

Assume that the tracked vehicle is running on a test course with a constant speed of 8 miles per hour (MPH). The test course consists of a flat, 100 foot segment with the exception of a bump between 50 and 60 feet, as shown in Fig. 11.

Table 3 shows the parameters used in the numerical integration for this example. Considered separately, the fast road wheel/arm subsystem has eigenvalues $\lambda_{\psi,1,2} = -116.65 \pm j9.056$, corresponding to an undamped natural frequency $\omega_{n,\psi} = 117$ rad/sec and

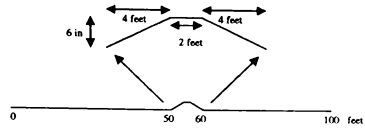


Figure 11. The test course of the tracked vehicle.

damping ratio $\zeta_\psi = 0.997$. The undamped natural frequencies in heave and pitch of the hull, when considered as a separate subsystem, are approximately 6.731 rad/sec and 1.946 rad/sec, respectively. Note that these are much lower than the 117 rad/sec natural frequency of each road wheel/arm subsystem.

Table 3. Parameter Values for the Tracked Vehicle Example.

parameter	value	parameter	value
m_h	3,600	m_i	15
I_h	180,000	I_i	6
k_{tb}	38,832	k_e	100,000
c_{sa}	3,333	c_e	100

When the function generation method is applied to this example, the fast subsystems (road wheel arms) are integrated off-line and the state variables at the next step, $\psi_{i,n+1}$ and $\psi_{i,n+1}$, are tabulated as functions of $\psi_{i,n}$, $\dot{\psi}_{i,n}$, and $T_{ne,n}$. Since all seven road arms have the same inertia properties, actually only one function table is needed for all of them. Without the subscript i , the functions become

$$\psi_{n+1} = f^\psi(\psi_n, \dot{\psi}_n, T_{ne,n}) \quad (30)$$

$$\dot{\psi}_{n+1} = f^{\dot{\psi}}(\psi_n, \dot{\psi}_n, T_{ne,n}) \quad (31)$$

In the above representation the motion of the chassis is assumed not to affect the dynamics of the road arms within each integration step. However, examination of the numerical size of each term in the equation shows that the velocity of the chassis in the normal direction of local terrain does have a significant effect on the torque T_{ne} because of its effect on the penetration rate \dot{s} . Thus the variable $v_{RN,n}$ is added to the function representation, where $v_{RN,n}$ represents the velocity of the road arm pivot point in the normal direction of local terrain at the beginning of the step. This again results in two four-variable functions, now defined as

$$\psi_{n+1} = f^\psi(\psi_n, \dot{\psi}_n, T_{ne,n}, v_{RN,n}) \quad (32)$$

$$\dot{\psi}_{n+1} = f^{\dot{\psi}}(\psi_n, \dot{\psi}_n, T_{ne,n}, v_{RN,n}) \quad (33)$$

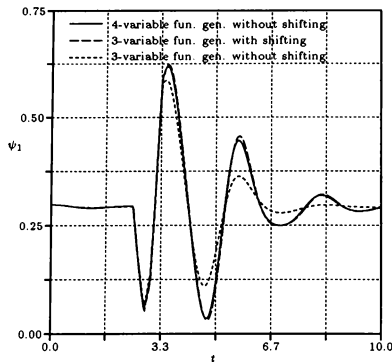


Figure 12. Simulation results with and without time shifting.

Fig. 12 shows the simulated time history of ψ_1 , the angle of road arm #1, as the tracked vehicle traverses the bump in Fig. 11 at 8 MPH. The four-variable function solution and the exact solution are essentially identical within the resolution of the graph. Fig. 12 shows that if the time shift scheme is applied in the case of three-variable function generation, the accuracy becomes comparable with that of the four-variable method. This is significant, since the three-variable function representation uses far less memory and requires less than one-half the computer execution time.

8. Conclusion

From the above examples it has shown that the function generation method is an efficient and stable method for handling stiff dynamic systems. The following cases represent examples for which the function generation method could be chosen over the multiple frame-rate method:

- Case 1: When the problem is extremely stiff, and the multiple frame-rate method requires a large frame ratio N to maintain stability.
- Case 2: When the problem has discontinuities, and the multiple frame-rate method requires a large frame ratio N to achieve acceptable accuracy.
- Case 3: When the fast subsystems are large and complex, and the time required for numerical integration is significant. Function generation moves the computationally intensive numerical integration to off-line calculation. This results

in much less on-line computation compared with the multiple frame-rate method, provided the number of input variables for the function generation method can be held within reasonable bounds.

9. Acknowledgment

The research reported in this paper has been partially funded by General Dynamics Land Systems Division.

References

- [1] Aiken, R. C., "Stiff Computation", *Proc. of the International Conference on Stiff Computation*, Park City, UT, April 1982, New York: Oxford Press.
- [2] Gear, C. W., "Simulation: Conflicts between Real-Time and Software", *Mathematical Software III*, Edited by Rice, J. R., Academic Press, 1977, pp. 121-138.
- [3] Gilbert, E. O., Howe, R. M., "An Expanded Role for Function Generation in Dynamic System Simulation", *Proc. 1977 Summer Computer Simulation Conference*, Chicago, July 1977, pp. 305-308.
- [4] Howe, R. M., "Dynamic Analysis of the State-Transition Method in Simulating Linear Systems", *Transactions of The Society for Computer Simulation*, Vol. 5, No. 1, January, 1988, pp. 27-41.
- [5] Howe, R. M., Haraldsdottir, A., "Multiple Frame Rate Integration", *Proc. of the AIAA Flight Simulation Technologies*, September 1988, pp. 26-35.
- [6] Kabamba, P. T., Yang, C., "On the Application of Generalized Sampled Data Hold Functions to Adaptive Systems with Controller Pre-Scheduling", *IEEE International Conference on System Engineering*, Dayton, August 1989.
- [7] Lin, K.-C., "The Use of Function Generation in the Real-time Simulation of stiff systems", Ph.D. Thesis, University of Michigan, 1990.
- [8] Palusinski, O. A., "Simulation of Dynamic Systems Using Multirate Integration Techniques", *Transactions of The Society for Computer Simulation*, Vol. 2, No. 4, December 1985, pp. 257-273.

SOME METHODS FOR REDUCING TIME DELAYS IN FLIGHT SIMULATION

R.M. Howe*

The University of Michigan
Ann Arbor, Michigan
Applied Dynamics International
Ann Arbor, Michigan

Abstract

Some methods for latency compensation that take advantage of the inherent lags in the dynamics of the airframe are examined in this paper. In particular, it is shown how rearranging the order of computation within each integration frame so that the calculations requiring the real-time inputs are left to last can lead to sizeable reductions in latency. This reduction in latency can be further improved by using a local linearization of the acceleration with respect to the inputs, where the required acceleration gradient is a byproduct of the acceleration function-generation mechanization. The dynamic performance of the simulation can also be improved through multi-rate input sampling, where the average of the samples over each frame is used in computing the acceleration associated with the frame. It is also shown how multi-rate input sampling and associated techniques can be combined to produce an accurate multi-rate output with almost no penalty in processing time. This in turn leads to much smoother and more accurate DAC outputs in an airframe simulation.

1. Introduction

It is well known that time delays associated with various elements involved in hardware-in-the-loop flight simulation can cause serious dynamic errors. These delays can be especially troublesome in man-in-the-loop simulators because of latency in CIG displays caused by pipelining in the display graphics algorithm, or because of dynamic lags inherent in motion systems.^{1,2} Although delays in the form of dynamic lags can be partially canceled by digital compensator formulas such as phase-lead algorithms, these methods do not alleviate the problem of pure latency, i.e., delays between cause and effect.³ In this paper we examine some methods for latency compensation that take advantage of the inherent lags in the dynamics of the airframe. To understand how these methods work we need to review the process of digital simulation using real-time integration methods. The state vector equations of motion for an airframe can be written in the following form:

$$\dot{V} = A[D, V, U(t)], \quad \dot{D} = V. \quad (1)$$

Here D , V , and A represent translational and rotational vectors of airframe displacement, velocity, and acceleration, respectively, and $U(t)$ is the real-time input vector, for example, control inputs by a pilot in a man-in-the-loop simulation. When the airframe is simulated in real time, a fixed integration step size h is used, since a variable step size is not compatible with real-time requirements. Thus the input and state vector components are represented at equally-spaced discrete times. The output

state vectors D_{n+1} and V_{n+1} at the $n+1$ frame, i.e., at $t = (n+1)h$, are computed from D_n and V_n at the n th frame using a real-time integration algorithm. To illustrate the way in which latency occurs in a real-time simulation, we consider next the numerical integration of Eq. (1) using a real-time algorithm.

2. Airframe Simulation Using AB-2 Integration

The most popular real-time algorithm is the second-order Adams-Bashforth predictor method known as AB-2. This is because AB-2 is a second-order method (dynamic errors proportional to h^2 , where h is the integration step size), requires only one evaluation of the state-variable derivatives per integration step, and is compatible with real-time inputs. When AB-2 integration is used, the difference equations which are solved by the simulation computer are given by

$$V_{n+1} = V_n + h \left(\frac{3}{2} A_n - \frac{1}{2} A_{n-1} \right), \quad (2)$$

$$D_{n+1} = D_n + h \left(\frac{3}{2} V_n - \frac{1}{2} V_{n-1} \right), \quad (3)$$

where

$$A_n = A(D_n, V_n, U_n). \quad (4)$$

At the beginning of the n th integration frame, i.e., at $t = nh$, the acceleration A_n is computed from the real-time input U_n , which has just become available, and the vectors D_n and V_n in accordance with Eq. (4). Then Eqs. (2) and (3) are used to calculate V_{n+1} and D_{n+1} . The simulation computer must be fast enough to complete all of these calculations in less than h seconds so that the states D_{n+1} and V_{n+1} , or any required function of these states, can be furnished as a real-time output at $t = (n+1)h$. It should be noted that in most airframe simulations the calculation of A_n in Eq. (4) requires much more computational time than the calculation of V_{n+1} and D_{n+1} in Eqs. (2) and (3). This is because A_n usually involves very complex nonlinear functions, including multivariable aerodynamic functions that are evaluated by table lookup and linear interpolation.

We now examine the latency problem associated with AB-2 integration. To do this we consider an input $U(t)$ which undergoes a sudden change and determine the elapsed time before this change is reflected in the simulation output displacement $D(t)$. We note first that the input $U(t)$ is sampled at integer frame times nh . This in turn means that the sample associated with a sudden change in $U(t)$ may be delayed for up to h seconds if the change occurs just after a previous sample. Eq. (2) shows that an additional one-frame delay occurs before the input sample U_n , which affects A_n , produces a change in the velocity V_{n+1} . Eq. (3) shows that this change shows up in the displacement D_{n+2} , a further one-frame delay. Thus the latency inherent in the AB-2 method when applied to the airframe equations given by (1) is between $2h$ and $3h$ seconds.

*Professor of Aerospace Engineering
Associate Fellow, AIAA

The above latency can be reduced by one frame (h seconds) if trapezoidal integration rather than AB-2 integration is used to obtain velocity from displacement. In this case Eq. (3) is replaced by

$$D_{n+1} = D_n + \frac{h}{2} (V_n + V_{n+1}) \quad (5)$$

Here V_{n+1} must be computed from Eq. (2) prior to computing D_{n+1} from Eq. (5). Clearly a change in V_{n+1} causes a change in D_{n+1} , instead of requiring a wait for D_{n+2} , as in Eq. (3). It also turns out that using trapezoidal rather than AB-2 to integrate velocity to displacement results in improved overall simulation accuracy, although the maximum allowable step size h compatible with numerical stability is reduced somewhat.⁴

3. Delay Compensation by Rearranging Computational Order

One method for reducing the latency effects described in the previous section is to complete as many of the calculations associated with the n th integration frame prior to nh , when the frame actually begins in real time. In mechanizing standard AB-2 integration, as represented by Eqs. (2), (3) and (4), there is no reason why Eq. (3) cannot be implemented at the end of the $n-1$ integration frame, since it depends only on V_n and V_{n-1} , both of which are available at the end of the $n-1$ frame. This means that the next displacement state, D_{n+1} , is available at nh , one entire frame ahead of real time. The 2 to 3 frame latency associated with conventional AB-2 integration is then reduced to between 1 and 2 frames. In an ongoing simulation this means that the output position state, within the accuracy of the AB-2 algorithm, is actually available h seconds ahead of when it would occur in real time. For example, this would permit exact compensation for at least h seconds of any delay associated with a CIG (Computer Image Generator) display. However, use of the trapezoidal algorithm in Eq. (5) achieves the same overall latency reduction for the combined airframe simulation and CIG system while maintaining a more accurate simulation because D_n is synchronized with respect to real time.

There may be other calculations required during the n th integration frame which can be completed prior to $t = nh$. In fact, the only calculations which cannot be started prior to time nh are those associated with the real-time input vector, U_n . Assume that these calculations take t_u seconds, and that the calculations which do not require U_n take $t_p = h - t_u$ seconds. Then the displacement D_{n+1} is available at time $t = (n+1)h - t_p$, i.e., t_p seconds ahead of real time. This procedure can indeed be used for exact compensation of t_p seconds of CIG or other delays, even when the trapezoidal method of Eq. (5) is also used for accuracy and latency improvement.

Unfortunately, in typical airframe simulations, much of the complexity associated with the nonlinear acceleration function in Eq. (4) involves the input U_n . For example, the pitch-axis component of acceleration depends on the aerodynamic pitching moment coefficient C_M . The coefficient C_M may in turn be a nonlinear function of angle of attack α , Mach number M , flap position δ_f , and elevator displacement δ_e , where δ_e is a component of the input vector U that results from control-stick displacement. The four-variable function representing C_M is evaluated each integration frame by table lookup and linear interpolation, a process that is computationally intensive. In a typical airframe simulation the time t_u required for calculation of

multi-variable functions involving real-time inputs can represent a substantial portion of the overall computation time for each integration step. As a result, the time $t_p = h - t_u$ available for advancing the output displacements in time, as described in the previous paragraph, may be somewhat limited.

However, there is a method to circumvent this limitation which leads to further potential improvements in the dynamic accuracy of simulations. In this method the algorithm for evaluation of multivariable functions involving real-time inputs is executed so that linear interpolation with respect to the real-time input is left to last. In the above example for the pitching moment coefficient C_M this means that interpolations with respect to α , M , and δ_f would be implemented first. This results in the calculation of the two quantities $C_M(\alpha, M, \delta_f, \delta_{ek})$ and $C_M(\alpha, M, \delta_f, \delta_{ek+1})$, where δ_{ek} and δ_{ek+1} are the data points within which the input δ_e falls. The final interpolation with respect to δ_e is accomplished using the formula

$$C_M(\alpha, M, \delta_f, \delta_e) = C_M(\alpha, M, \delta_f, \delta_{ek}) + [C_M \delta_e]_k (\delta_e - \delta_{ek}) \quad (6)$$

where

$$[C_M \delta_e]_k = \frac{C_M(\alpha, M, \delta_f, \delta_{ek+1}) - C_M(\alpha, M, \delta_f, \delta_{ek})}{\delta_{ek+1} - \delta_{ek}} \quad (7)$$

Implementation of Eq. (6) requires only two additions and one multiplication, assuming $[C_M \delta_e]_k$ has already been calculated. Of course, knowledge of the real-time input δ_e is necessary to determine the function table addresses for the data needed to accomplish the first three interpolations with respect to α , M , and δ_f , respectively, and hence to compute $[C_M \delta_e]_k$. But suppose an estimate, $\hat{\delta}_e$, as might be obtained from extrapolation from previous δ_e values, is used to compute δ_{ek} . This permits the computationally intensive calculation of $[C_M \delta_e]_k$ to proceed without waiting for the real-time input δ_e . When the real-time δ_e finally arrives, only two additions and one multiplication, as noted above, are then required to complete the computation of the coefficient C_M in Eq. (6) and hence the pitch component of angular acceleration. This procedure is essentially equivalent to a local linearization of the acceleration function with respect to the input, where the linearization coefficient is updated each integration frame. As noted above, the procedure postpones the time when the real-time input is needed until t_u seconds before the end of the computational frame, where t_u can be small compared with the integration step size h . This in turn makes the frame output displacement available $h - t_u = t_p$ seconds ahead of its occurrence in real time.

4. A Modified Form of AB-2 Integration

The procedure for local linearization of the acceleration function with respect to the input, as described in the previous section, can be used to further reduce the latency as well as improve the accuracy of a real-time simulation. In order to understand how this is accomplished, it is useful to introduce a modified form of the AB-2 predictor algorithm. In this method the acceleration vector A is evaluated at time $t = (n+1/2)h$ instead of nh , as in Eq. (4). To do this it is necessary to have estimates of the velocity V , displacement D , and input U at the $n+1/2$ frame. These can be obtained by linear extrapolation using the n and $n-1$ frame values. Thus we let

$$\begin{aligned}\hat{V}_{n+1/2} &= \frac{3}{2} V_n - \frac{1}{2} V_{n-1}, \quad \hat{D}_{n+1/2} = \frac{3}{2} D_n - \frac{1}{2} D_{n-1}, \\ \hat{U}_{n+1/2} &= \frac{3}{2} U_n - \frac{1}{2} U_{n-1}.\end{aligned}\quad (8)$$

The difference equations for solving Eq. (1) are now given by

$$V_{n+1} = V_n + hA(\hat{D}_{n+1/2}, \hat{V}_{n+1/2}, \hat{U}_{n+1/2}) \quad (9)$$

$$D_{n+1} = D_n + h\hat{V}_{n+1/2} \quad (10)$$

If the acceleration A is a linear function of D , V , and U , it is evident that this modified form of AB-2 integration is exactly equivalent to the standard AB-2 method represented by Eqs. (2), (3), and (4). However, when A is a nonlinear function of D , V , and U , which is invariably the case in flight equations, the two methods give different results. In standard AB-2 integration, first order extrapolation is applied to the derivatives of the state variables, whereas in the above modified AB-2 method, the first-order extrapolation is applied to the states. If the acceleration A contains discontinuous nonlinear functions, or if the input $U(t)$ has high frequency content, then the velocity state V will be a smoother function of time than the acceleration A . Under these conditions, which often occur, extrapolation based on the states instead of the state derivatives will give better results, and the modified AB-2 method should be more accurate than standard AB-2. This indeed turns out to be true in typical nonlinear airframe and control system simulations.

Of course even better accuracy can be achieved by using $U_{n+1/2}$, the actual input at the $n+1/2$ frame, instead of the estimate $\hat{U}_{n+1/2}$ given in Eq. (8). This does mean that an additional half-frame wait must occur before $U_{n+1/2}$ is available as a real-time input. But use of the technique of rearranging the order of computation and/or the local linearization of the acceleration function, as described in Section 3, should still permit the output state D_{n+1} to be computed in real time, or perhaps even slightly ahead of real time.

Yet a more accurate method for treating the input $U(t)$ in Eq. (9) is to utilize the average value of the input over the n th frame, $\bar{U}_{n,n+1}$, in place of $\hat{U}_{n+1/2}$ in Eq. (9). In fact, for the case where the acceleration A is simply equal to the input U (no dependence on D or V), use of the average input, $\bar{U}_{n,n+1}$, in Eq. (9) will give an exact result for V_{n+1} . This is because the average, $\bar{U}_{n,n+1}$, times the step size h is indeed the area under the U versus t curve. A numerical approximation to this average can be obtained by averaging N samples of the input U over the time interval from nh to $(n+1)h$. Thus we use the formula

$$\begin{aligned}\bar{U}_{n,n+1} &= \frac{1}{N} [U_{5/N} + U_{1.5/N} + U_{2.5/N} + \cdots + U_{(N-5)/N}] \\ \text{or} \quad \bar{U}_{n,n+1} &= \frac{1}{N} \sum_{k=1}^N U_{(k-1/2)/N}\end{aligned}\quad (11)$$

Figure 1 illustrates how Eq. (11) approximates the area under $U(t)$ versus t over one integration frame. Of course, when this method of multi-rate input sampling is used, the computation of $\bar{U}_{n,n+1}$ cannot be completed until the last real-time input sample arrives, namely at $t = (n+1-1/2N)h$. This leaves only $1/2N$ seconds to complete the computation of the real-time output for the $n+1$ frame. It is still possible to do this by rearranging the

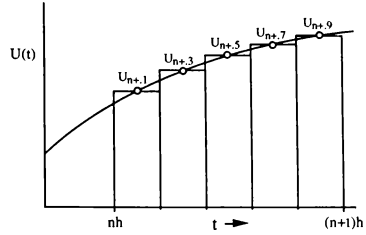


Fig. 1. Numerical approximation to the average of $U(t)$ over one frame using 5 samples ($N=5$).

computation order and using the scheme of local linearization of acceleration with respect to the input, as described in Section 3.

If N , the number of input samples averaged over each frame, is reasonably large, the latency variability of zero to h seconds associated with one input sample per frame is virtually eliminated. When combined with the modified AB-2 algorithm followed by the trapezoidal method for integrating acceleration to obtain velocity and displacement, respectively, this reduces to zero the overall input-to-output latency, compared with $2h$ to $3h$ seconds when conventional AB-2 integration is used.

5. Prediction Using Modified Euler Integration

In the modified AB-2 method introduced in the previous section we used linear extrapolation to estimate the states and input at the half-integer frame time with Eq. (6). These half-integer values were then used in the modified Euler formulas of Eqs. (9) and (10). The extrapolation formulas in Eq. (8) and the attendant errors can be avoided, at least partially, by computing the velocity states at half-integer frame times. The displacement states and accelerations are still represented at integer states. In this case the difference equations for solving Eq. (1) become

$$V_{n+1/2} = V_{n-1/2} + hA(D_n, \hat{V}_n, U_n), \quad (12)$$

$$D_{n+1} = D_n + h\hat{V}_{n+1/2}. \quad (13)$$

Here the only required extrapolation is that needed to estimate V_n , since the velocity state is now only defined at half-integer times. One method for doing this is to use first-order extrapolation based on $V_{n-1/2}$ and $V_{n-3/2}$. In this case

$$\hat{V}_n = \frac{3}{2} V_{n-1/2} - \frac{1}{2} V_{n-3/2} \quad (14)$$

As noted earlier, this is equivalent to AB-2 integration for any terms in the acceleration function A which involve the velocity state V . The remainder of the terms are effectively integrated with the modified-Euler method, which exhibits dynamic errors that are approximately 10 times smaller than those associated with AB-2 integration.⁵ A second method for computing the estimate of V_n is to use second-order prediction integration based on A_{n-1} and A_{n-2} . In this case the formula is given by

$$\hat{V}_n = V_{n-1/2} + h(\frac{7}{8}A_{n-1} - \frac{3}{8}A_{n-2}) \quad (15)$$

Since the local truncation error in computing \hat{V}_n using Eq. (14) is proportional to h^3 , the full global accuracy of order h^2 associated with modified Euler integration will be preserved. The only disadvantage in using Eq. (14) is the reduction in the maximum allowable step size h before numerical instability occurs.⁶ If this is a problem, trapezoidal integration can be used by factoring V out of the acceleration term and solving explicitly for $V_{n+1/2}$.⁷

At the end of Section 3 we noted that it is possible to compute the output displacement, D_{n+1} , t_p seconds ahead of when it occurs in real time, where t_p can be almost as large as the integration step size h . This was accomplished using the combined AB-2/trapezoidal integration algorithms as well as the rearranged computation order and local linearization of acceleration with respect to the input. If the modified-Euler method of Eqs. (12), (13) and (15) is used instead of AB-2/trapezoidal integration, then nearly one full frame of output prediction can be achieved accompanied by almost an order of magnitude improvement in dynamic accuracy.

The modified Euler method is of course also compatible with the multi-rate input sampling and averaging represented by Eq. (11). Reference to Eq. (12) shows that in this case we need to compute $\bar{U}_{n-1/2, n+1/2}$, i.e., the average value of the input over the interval $(n-1/2)h, (n+1/2)h$. For N samples per integration step the formula becomes

$$\bar{U}_{n-1/2, n+1/2} = \frac{1}{N} \left[U_{(1-N)/2N} + U_{(3-N)/2N} + \dots + U_{(N-1)/2N} \right] \quad (16)$$

or

$$\bar{U}_{n-1/2, n+1/2} = \frac{1}{N} \sum_{k=1}^N U_{(2k-1-N)/2N}$$

U_n in Eq. (12) is replaced by $\bar{U}_{n-1/2, n+1/2}$ to implement the multi-rate input sampling. As in the case of AB-2 integration, this yields a further improvement in dynamic accuracy and at the same time removes the uncertainty in latency associated with a single sample per integration frame. Here the maximum prediction interval that can be attained when using the techniques of Section 3 is $h/2$ seconds, since in the n th integration frame it is now necessary to wait almost until $t = (n+1/2)h$ to complete the calculations involving the input U .

6. Use of Multi-rate Inputs and Outputs

Multi-rate input sampling can be combined with local linearization of the acceleration function, as described in Section 3, to provide an output data sequence at a frame rate which is a multiple N of the basic integration frame rate used in the airframe simulation. This may be useful in driving output DAC's (Digital-to-Analog Convertors) in a hardware-in-the-loop simulation in order to minimize the jumps in DAC output from one update to the next. The multi-rate output data sequence can be obtained by simple numerical integration formulas if it is assumed that during each frame interval h , the acceleration only changes as a result of changes in the input. To illustrate how the scheme works, we consider again the basic airframe equations given by (1). Assume also that we use the modified AB-2 formulation of Section 4 in order to take advantage of multi-rate input sampling and averaging, as

represented in Eq. (11). At the beginning of the n th frame in real time, we have just finished computing V_n using the formula

$$V_n = V_{n-1} + h A_{n-1/2} \quad (17)$$

Here the acceleration $A_{n-1/2}$ is, by analogy with Eqs. (6) and (7), given by

$$A_{n-1/2} = A(\hat{D}_{n-1/2}, \hat{V}_{n-1/2}, \hat{U}_{n-1/2}) + A_{U_{n-1/2}} \cdot (\bar{U}_{n-1/2} - \hat{U}_{n-1/2}), \quad (18)$$

where $A_{U_{n-1/2}}$ represents the partial derivative of the acceleration A with respect to the input U . As noted in Eqs. (6) and (7), this is a byproduct of the table lookup and linear interpolation algorithm for computing $A(D, V, U)$. If the trapezoidal method is used for integrating V to obtain D , then at the beginning of the n th frame we have also just finished computing D_n from the formula

$$D_n = D_{n-1} + \frac{h}{2} (V_n + V_{n-1}) \quad (19)$$

To obtain N output samples over the n th frame, we start by computing the output displacement $D_{n+5/N}$ with the integration formula

$$D_{n+5/N} = D_n + \frac{5h}{N} V_n \quad (20)$$

This is simply Euler integration with a step size of $5h/N$.

Next we use modified Euler integration to compute $V_{n+1/N}$. To do this we need the acceleration $A_{n+5/N}$. If we assume that this acceleration only differs from $A_{n-1/2}$ because of the change in input U , then by analogy with Eq. (18) we can write

$$A_{n+5/N} = A_{n-1/2} + A_{U_{n-1/2}} \cdot (U_{n+5/N} - \bar{U}_{n-1/2}) \quad (21)$$

Then $V_{n+1/N}$ is given by

$$V_{n+1/N} = V_n + \frac{h}{N} A_{n+5/N} \quad (22)$$

From this result we compute the next output displacement, $D_{n+1.5/N}$, from the modified-Euler formula

$$D_{n+1.5/N} = D_{n+5/N} + \frac{h}{N} V_{n+1/N} \quad (23)$$

This is followed by the calculation of the acceleration $A_{n+1.5/N}$, which is given by

$$A_{n+1.5/N} = A_{n-1/2} + A_{U_{n-1/2}} \cdot (U_{n+1.5/N} - \bar{U}_{n-1/2}) \quad (24)$$

This in turn is used to compute $V_{n+2/N}$ and $D_{n+2.5/N}$ in the next modified-Euler integration step. Thus

$$V_{n+2/N} = V_{n+1/N} + \frac{h}{N} A_{n+1.5/N} \quad (25)$$

and

$$D_{n+2.5/N} = D_{n+1.5/N} + \frac{h}{N} V_{n+2/N} \quad (26)$$

In this way the multi-rate integrations are continued for a total of N steps to produce $V_{n+1/N}, V_{n+2/N}, \dots, V_{n+(N-1)/N}$, and $D_{n+5/N}, D_{n+1.5/N}, D_{n+2.5/N}, \dots, D_{n+(N-5)/N}$. Each integration step uses the latest real-time input sample. Since each step requires a total of only 4 additions and 3 multiplications per

scalar component, it should be possible for the calculations to keep up with real time. The simplicity of the equations even suggests that they can be performed with a microprocessor chip located at the real-time input/output interface. This interface microprocessor can also be used for on-line calculation of the average input, $\bar{U}_{n,n+1}$, over the n th frame, as given in Eq. (11). At the beginning of each frame of multi-rate integrations the main processor passes $D_n, V_n, A_{n-1/2}$, and $A_{n-1/2}$ to the interface processor. As the real-time, multi-rate input samples are taken, the interface processor solves the above modified-Euler integration equations to produce the multi-rate, real-time outputs. These outputs can in turn be used to drive DAC's, including compensation for the half-frame delay associated with the zero-order DAC outputs. This procedure will be illustrated with an example simulation in the next section. After the N multi-rate integrations over the n th frame are completed by the interface processor, it passes the average input, $\bar{U}_{n,n+1}$, back to the main processor, where it is used to compute the final and accurate values of the next states, V_{n+1} and D_{n+1} .

It should be noted that D, V , and A in the above equations are vectors, whereas the computations in an actual simulation involve equations based on the scalar components of these vectors. In general the scalar components of the displacement vector D will be defined along different axes than the components of the velocity vector V . For example, the translational and rotational velocity components may be defined along body axes, whereas the displacement components may be defined along earth or inertial axes. In this case it is necessary to mechanize an axis transformation to convert components of the velocity vector V from body to inertial axes prior to integrating them to obtain the displacement components. For each multi-rate integration step this will require additions and multiplications involving the appropriate direction cosines. The computational load is considerably reduced if it is assumed that the direction cosines relating the two axis systems can be considered constant over each main integration frame time, h . It should be remembered that any errors caused by assumptions such as this only produce local errors in the multi-rate outputs. Each main integration step the states are restored to the full accuracy associated with the algorithm used for the integrations with step-size h .

In developing the above multi-rate integration formulas, we have assumed that the acceleration A over the n th frame differs from its value $A_{n-1/2}$ at the midpoint of the previous frame only as a result of input changes. Although this assumption gives good results, as the example simulation in the next section will show, it will obviously causes dynamic errors in the multi-rate outputs, especially if the dominant airframe transients vary significantly over the step-size h . There are various ways in which the accuracy of the approximation for A over the n th frame can be improved. One obvious way is to use extrapolation based on $A_{n-1/2}$ and $A_{n-3/2}$ to determine an estimate for $A_{n+1/2}$. This is then corrected at each integration substep during the n th frame for the difference between the actual real-time input and the extrapolated input $\bar{U}_{n+1/2}$ given by $2\bar{U}_{n-1,n} - \bar{U}_{n-2,n-1}$.

The above development of multi-rate integration formulas based on the use of multi-rate inputs and outputs has assumed that the airframe equations of motion given by (1) are integrated using the AB-2/trapezoidal scheme. We have noted in Section 5 that the modified-Euler integration method can produce up to an order of magnitude improvement in accuracy when used to

integrate the airframe equations of motion with the same step size h . This method can of course also be combined with the multi-rate input/output scheme within each frame. In this case the multi-rate integration formulas, as described above, are modified somewhat because modified-Euler integration of the main airframe equations produces output displacements at integer frame times and output velocities at half-integer frame times.

7. Example Simulation

In this section we consider an example simulation to illustrate the performance of the various methods described in the paper. We choose a linear, second-order system for its simplicity and because it is representative of such typical airframe dynamic behavior as the short-period and Dutch-roll transients associated with longitudinal and lateral motion, respectively. Thus the equations of motion to be simulated are the following:

$$\dot{V} = A, \quad \dot{D} = V, \quad (27)$$

where

$$A = \omega_n^2(U-D) - 2\zeta\omega_n V. \quad (28)$$

Here ω_n is the undamped natural frequency and ζ is the damping ratio of the second-order system. As before, A represents acceleration, and D and V represent velocity and displacement, respectively.

As a test input we consider an acceleration-limited unit step function, where the input acceleration is given by

$$\begin{aligned} \ddot{U} &= \frac{1}{T^2}, \quad 0 \leq t < T, & - \\ &= -\frac{1}{T^2}, \quad T \leq t < 2T, & (29) \\ &= 0, \quad t \geq 2T \end{aligned}$$

When integrated twice with T set equal to $1.2/\omega_n$, Eq. (29) yields the input shown in Figure (2). The ideal second-order system response for $\zeta = 0.25$ and initial conditions $D(0) = V(0) = 0$ is also shown in the figure. The response is very nearly the same as that obtained when a true step input is applied at T , i.e., at $\omega_n t = 1.2$ in Figure 2. Use of the acceleration-limited step input removes the discontinuities in displacement and slope

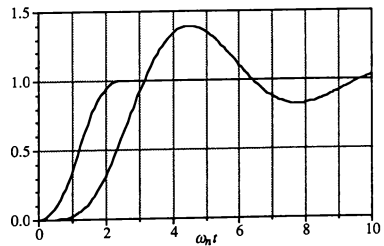


Fig. 2. Acceleration-limited step input and second-order system response; $\zeta = 0.25$.

which occur in an ideal step input. These discontinuities cause large transient errors when predictor algorithms such as AB-2 are used for numerical integration. The acceleration-limited step input is also a more realistic input for a physical system such as an airframe.

Figure 3 shows the solution errors in simulating the acceleration-limited second-order system response using the different integration methods described in the paper. The integration step size h for these simulations was set equal to $0.4/\omega_n$, i.e., $\omega_n h = 0.4$. This corresponds to 2.5 integration steps per radian or about 15 steps per cycle of the transient solution. The figure shows that when the AB-2 method is used for both integrations, the largest errors are generated. Replacing AB-2 with the trapezoidal method for the integration of velocity V to obtain the displacement D makes a substantial improvement in dynamic accuracy. Use of the modified-Euler method, with AB-2 integration for the V -dependent damping term, shows a further improvement in accuracy. Finally, the modified-Euler with the predictor damping, as represented by Eq. (15), exhibits by far the smallest errors of any of methods shown. Both modified-Euler cases in Figure 3 use the input

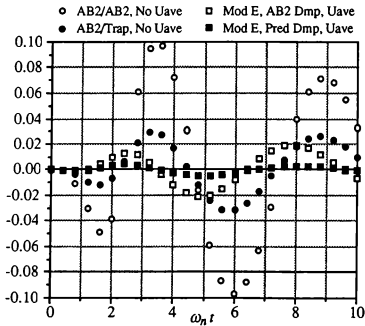


Fig. 3. Step-response errors of different second-order integration algorithms; $\omega_n h = 0.4$.

average each frame based on Eq. (16) with N samples per frame. Clearly the modified-Euler method with predictor damping and multi-rate input averaging provides more than an order of magnitude improvement in accuracy when compared with the conventional real-time AB-2 method of integration. Similar accuracy improvements have been observed for modified-Euler integration in simulating the complete nonlinear flight equations.⁷ This is significant, since it means that for a given accuracy, the modified-Euler method permits the integration step size to be increased by a factor of approximately three. The utilization of multi-rate input-output samples, as described in Section 6, eliminates large latency uncertainties and preserves good input/output response dynamics even when using much larger integration steps.

We assume next that the output displacement D is used to drive a DAC in order to generate a simulated continuous output.

Figure 4 shows a plot of the DAC output when conventional AB-2 integration is used in simulating the acceleration-limited step response of the second-order system. Again $\zeta = 0.25$ and $\omega_n h = 0.4$. Also shown in the figure is the exact continuous solution. It can be seen that the simulated output errors are substantial, as are the discontinuous jumps in the DAC output

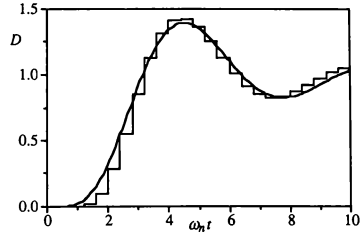


Fig. 4. DAC output with convention AB-2 integration used to simulate the step response; $\omega_n h = 0.4$. Also shown is the exact continuous solution.

resulting from the relatively large step size. Figure 5 shows the DAC output when the trapezoidal method is used instead of the AB-2 algorithm in integrating velocity V to obtain displacement D . Clearly the combined AB-2/trapezoidal integration reduces significantly the errors in the output data points D_n , i.e., at the time each DAC update occurs. But the half-frame delay associated with the DAC still produces a substantial equivalent error in the continuous DAC output. This delay can be compensated by driving the DAC with an extrapolated output value $\hat{D}_{n+1/2}$, which represents a time advance of $h/2$ seconds. For linear extrapolation the formula is simply

$$\hat{D}_{n+1/2} = \frac{3}{2}D_n - \frac{1}{2}D_{n-1} \quad (30)$$

Figure 6 shows the DAC output in this case. The elimination of the half-frame delay is quite evident, although the same discontinuous jumps in DAC output are still present.

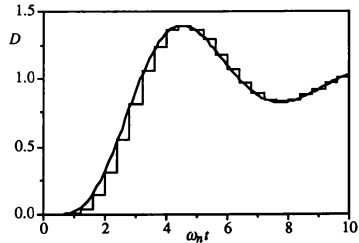


Fig. 5. DAC output when combined AB-2/trapezoidal integration is used to simulate the step response; $\omega_n h = 0.4$.

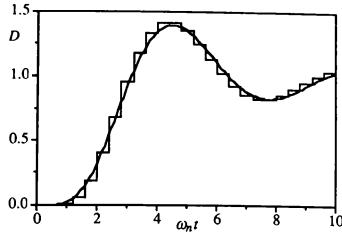


Fig. 6. Delay-compensated DAC output when combined AB-2/trapezoidal integration is used to simulate the step response; $\omega_n h = 0.4$.

We next consider use of multi-rate inputs and outputs, as described in Section 6. We let $N = 4$. This means that the input U is sampled 4 times during each main integration frame of duration h , namely, at $(n+1/8)h$, $(n+3/8)h$, $(n+5/8)h$ and $(n+7/8)h$. The multi-rate output D is obtained for these same times using the multi-rate integration scheme of Section 6. The DAC being driven by the multi-rate output data sequence is updated during each frame at $t = nh$, $(n+1/4)h$, $(n+2/4)h$ and $(n+3/4)h$, respectively, i.e., with a lead time of $h/8$. Again, this compensates for the effective half-frame delay in the multi-rate DAC output. Figure 7 shows the DAC output when combined AB-2/trapezoidal integration is used for the main integration with step-size $h = 0.4\omega_n h$ as before. Comparison with Figure 6 shows how much smoother the DAC output becomes, even though we have only used 4 multiple samples per frame.

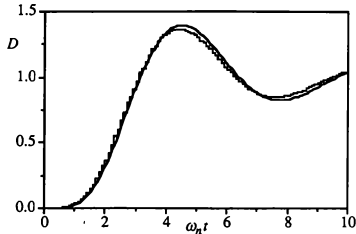


Fig. 7. Delay-compensated DAC output when combined AB-2/trapezoidal integration is used as well as multiple I/O sampling ($N=4$); $\omega_n h = 0.4$.

Figure 8 shows the multi-rate DAC output when modified-Euler with predictor damping is used for the main integration. Again, $N = 4$. To more effectively show the improved accuracy of the modified-Euler method we have magnified the vertical axis by a factor of 2. The comparison of Figures 4 and 8 shows how much the use of modified-Euler integration with multi-rate input-output sampling instead of conventional AB-2 integration has improved the fidelity of the simulated continuous output.

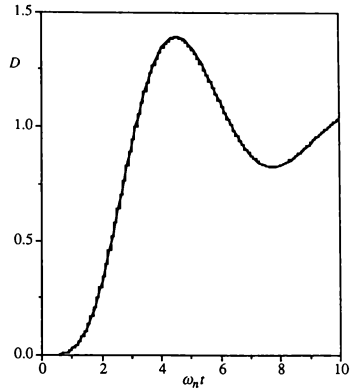


Fig. 8. Delay-compensated DAC output when modified-Euler integration is used as well as multiple I/O sampling ($N=4$); $\omega_n h = 0.4$.

8. Conclusions

In this paper we have shown that the latency associated with conventional AB-2 integration can be reduced by h , the integration step size, by computing and outputting the displacement D_{n+1} for the n th integration frame at the end of the $n-1$ frame, that is, immediately after the velocity V_n has been calculated. An alternative method for latency reduction by h seconds is to use the trapezoidal method instead of the AB-2 algorithm in integrating velocity V to obtain displacement D . This has the added advantage of improving the overall accuracy of the simulation. We have also shown how additional reduction in latency can be accomplished by rearranging the order of computation within each integration frame so that the calculations requiring the real-time input are left to last. This reduction can be further improved by using a local linearization of the acceleration with respect to the input, where the required acceleration gradient is a byproduct of the acceleration function-generation mechanization. The dynamic performance of the simulation can be further improved through multi-rate input sampling, where the average of the samples over each frame is used in computing the acceleration associated with the frame. Optimal use of all of the above techniques can reduce the latency of an AB-2/AB-2, or AB-2/trapezoidal simulation to zero. We have shown how by applying these techniques along with modified-Euler integration, we obtain not only improved dynamic accuracy but also an effective output lead of $h/2$ seconds. Finally, we have shown how the multi-rate input sampling and associated techniques can be combined to produce an accurate multi-rate output with almost no penalty in processing time. This in turn leads to much smoother and more accurate DAC outputs in an airframe simulation.

References

1. Levison, W.H., and B. Papazian, "The Effects of Time Delay and Simulator Mode on Closed-loop Pilot/Vehicle performance: Model Analysis and Manned Simulation Results," *Proc. of the AIAA Flight Simulation Technologies Conference*, Aug. 17-19, 1987, pp 39-49.
2. Merriken, M.S., G.E. Riccio, and W.V. Johnson, "Temporal Fidelity in Aircraft Simulator Visual Systems," *Proc. of the AIAA Flight Simulation Technologies Conference*, Aug. 17-19, 1987, pp 50-54.
3. Sobiski, D.J., and F.M. Cardullo, "Predictive Compensation of Visual System Time Delays," *Proc. of the AIAA Flight Simulation Technologies Conference*, Aug. 17-19, 1987, pp 59-70.
4. Howe, R.M., "The Use of Mixed Integration Algorithms in State Space," *Transactions of the Society for Computer Simulation*, 7 (1): 45-66.
5. Howe, R.M., "Simulation of Linear Systems Using Modified Euler Integration Methods," *Transactions of the Society for Computer Simulation*, 5 (2): 125-152.
6. Howe, R.M., "A Performance Comparison of Integration Algorithms in Simulating Flexible Structures," *Proc. of the NASA Workshop on Computational Aspects in the Control of Flexible Systems*, July 12-14, 1988 Williamsburg, VA.
7. Howe, R.M., "An Improved Numerical Integration Method for Flight Simulation," *Proc. of the AIAA Flight Simulation Technologies Conference*, Aug. 14-16, 1989, pp 310-316.

OPTIMIZING THE MPX OPERATING SYSTEM FOR
MULTI-TASKING REAL-TIME PROGRAMS

Laird A. Parker*
Data Systems Division

Flight Simulation Laboratory
Fort Worth Division
General Dynamics Corporation
P. O. Box 748, MZ 5945
Fort Worth, TX 76101

Abstract

After discerning the internals of MPX** (Encore Operating System), a new system module, H.SIMS, was designed to reduce the context switch time for IPU-biased real-time tasks. Careful timing tests comparing H.SIMS to the MPX context switch service, H.SURE, indicate that the custom module dramatically achieved its goal, making an IPU context switch only slightly longer than a CPU context switch.

MPX and the IPU

The Flight Simulation Laboratory at General Dynamics Fort Worth Division uses several Encore 32 97/80 dual processor machines in a heavily multi-tasked real-time environment. In the fall of 1988, I was tasked to determine how effectively the IPU (internal processing unit, similar to CPU) was then being utilized and whether this could be improved. Assembled source listings of the following MPX modules were acquired: H.EXEC, H.CPU, H.IPU, H.IPO6, and H.SURE. The first step was to completely understand the MPX internals associated with context switching. The only way to truly understand internals is to understand the source, and one pass through the source is not nearly enough.

MPX was originally designed and written for a single processor computer, predating any notion of an IPU. The module, H.EXEC, contains all of the routines for maintaining the DQE (dispatch queue entry) state queues and the most important routine, S.EXEC20, the dispatcher which performs CPU context switching. When the IPU was invented, MPX was patched to accommodate it. H.EXEC was modified and H.CPU and H.IPU were added. The two new modules each handle the trap generated when the opposite processor executes an SIPU (signal IPU) instruction. In addition, H.IPU contains code to process all of the traps generated in the IPU and, in its entirety, the IPU operating system, as the IPU never executes any MPX code not found within H.IPU.

Whenever a state queue, the real-time ready queue, SQR, for instance, is being modified, interrupts are blocked by H.EXEC to prevent havoc. Since blocking interrupts in the CPU has no effect whatsoever on the IPU, this method will not prevent the IPU from attempting to modify an MPX data structure, such as a state queue, while the CPU is simultaneously doing the same thing. So, therefore, it never tries to do so. This is why the IPU cannot context switch itself.

*Computer Systems Engineering Specialist

**MPX-32 is a trademark of Encore Computer Corporation

Whenever H.IPU determines that it cannot continue executing the CIPU (current IPU state) task, it simply traps the CPU and halts (an SIFU followed by WAIT instruction pair). Now the CPU must stop what it is doing, and H.CPU will switch the halted IPU task out of CIPU and into some ready queue, then attempt to switch some other task into CIPU and wake the IPU up with an SIFU instruction. The point is that with the addition of the IPU, MPX was not modified to allow the IPU to touch a single MPX data structure. This could have been done with spin locks and some careful thought.

What of the IPU scheduling algorithm itself (as implemented through the changes to H.EXEC and the addition of H.CPU)? It is a general-purpose algorithm that must work in both real-time and non-real-time, and makes no distinction between the two environments. In real-time, you cannot afford for a task to make multiple switches between the CPU and IPU in a single frame, although MPX leaves this possibility wide open. Additionally, none of the MPX utilities such as the Assembler, FORTRAN Compiler, Cataloger, or Volume Manager are built "cpuonly", which means that if you are running real-time programs, these MPX pests will all get into the IPU whenever they can, forcing the real-time programs to wait a longer amount of time to gain the IPU once resumed. Of course, in a development environment, you do benefit from the IPU being able to execute the MPX utilities (although I have my doubts about Volume Manager). MPX includes no utilities to monitor the IPU performance, except the OPOOM "list IPU" command, which is woefully inadequate because the 20 entries are not necessarily in a non-interrupted temporal sequence.

Let us run through a possible task scheduler scenario under MPX. You have four real-time programs. Two are built "cpuonly", while the other two are built "ipubias". Despite the fact that no MPX documentation ever suggests it,

you know it is best for both of your IPU tasks to be higher priority than your CPU tasks. An interrupt handler resumes all four of your tasks. The first IPU task executes briefly and then does any one of the forbidden things such as an SVC (system service) instruction. This will trap the CPU in H.CPU, which will schedule the second IPU task for the IPU and return through the dispatcher, which will switch out the current CPU task and switch in the first IPU task (since it is higher priority than the current CPU task). The first IPU task, now in the CPU, will reexecute the offending SVC instruction. All SVC's return through the dispatcher, and when this happens, the dispatcher will switch the current CPU task (the first IPU-biased task) out of the CPU and into the IPU-ready state, then force the IPU to halt the current (second) IPU task (even though it will have only executed a few instructions by that point), and finally select a task (the CPU-only task that was originally interrupted) to run in the CPU. Before the CPU task executes any instructions, however, the H.CPU trap handler will be entered in order to switch out the second IPU task (which did not have the IPU long enough to do much) in order to switch the first IPU task back into the IPU. Having done this, H.CPU will again return through the dispatcher which will determine that all is well and finally return to the CPU task that was first interrupted so long ago. If, at this moment, the first IPU task issues another illegal IPU instruction, the nightmare continues.

The lessons learned should be: 1) any task that you do not want running in the IPU must be built CPU-only--the default scheduler treats IPU-biased tasks and unbiased tasks about the same, 2) if an IPU-biased task ever traps the CPU other than to suspend itself, it should be made CPU-only, 3) the OPOOM "list IPU" command can tell you if an IPU task is trapping the CPU, but cannot necessarily tell you that it isn't,

and 4) all IPU-biased tasks should be higher priority (lower number) than all CPU-only tasks.

MPX Customized for Real-Time

Rather than modify any MPX source and rebuild MPX object, a new system module, H.SIMS, was designed to be linked (compressed) into the MPX image. H.SIMS contains an interrupt handler for servicing a periodic top-of-frame interrupt as well as several SVC entry points, one of which is called by real-time tasks to suspend themselves. Another SVC is called once, by an initialization program to set up the interrupt handler and cause H.SIMS to assume control of the IPU. One of the problems with MPX is that it makes no distinction between using the IPU for real-time or for development. Once H.SIMS takes control of the IPU, not even IPU-biased tasks will get the IPU unless they use the H.SIMS services. H.SIMS insures that all IPU tasks (up to ten) are higher priority than all CPU tasks (also up to ten), and will execute the tasks in the standard order of decreasing priority after the top-of-frame interrupt.

H.SIMS detects whenever an IPU task traps the CPU and, through an interactive utility, reports this to the user. Once an IPU task has trapped to the CPU, H.SIMS will insure that it stays in the CPU for the remainder of that frame (until it calls the H.SIMS context switch SVC). The next frame it will be resumed in the IPU. This makes it impossible for a task to thrash back and forth between the IPU and CPU, since it can only trap once per frame.

When the real-time programs are no longer needed, another SVC is called to disable the interrupt handler and restore the IPU to the default MPX scheduler. The system will now operate normally. So, without rebooting, H.SIMS can switch the IPU from the normal mode, to a real-time only mode, and back again. Even when in real-time mode, the CPU scheduler operates

exactly the same as normal so that non-real-time tasks can execute in the CPU (but not in the IPU) when H.SIMS is in real-time mode.

H.SIMS vs. H.SURE Timing Tests

The new context switching module, H.SIMS, was compared to the MPX module, H.SURE, in a series of 34 timing tests. Three events were timed: 1) CPU context switch time, 2) IPU context switch time, and 3) the time of CPU task interruption during an IPU context switch. Each event was tested under a variety of cache conditions. The results clearly indicate that the practice of quoting a single time for the execution of a context switch on a machine incorporating cache is virtually useless. For the two context switch events, six unique tests were performed with different cache conditions. For the third event, five unique tests were performed.

Testing Environment

All tests were performed on the same Encore 32 97/80 computer. While the tests were in progress, the entire machine was dedicated to the testing effort. All background real-time utilities (for file transfer, system performance monitoring, etc.) were removed prior to testing. There were no interactive users logged on during testing except when required by the tests, as will be explained. The operating system was MPX 3.4 U02, and was built with both H.SURE and H.SIMS so that all tests were performed with the same system image. The default scheduler (H.EXEC and H.CPU) was used.

Timing Method

All times were measured using the Bus Performance Monitor (BPM) manufactured by Custom Computer Products Corporation. This product is a SEIbus board that directly drives an RS232 terminal. The board never

accesses the bus and is, therefore, entirely non-intrusive. The BPM is a bus monitor, or snooper, capable of measuring the time between two bus events which are each defined by address, data, and transaction type. Simple FORTRAN assignment statements were used for the context switch time start and stop events. By assigning a literal constant to a variable whose physical address is known, the BPM can be set up to trigger upon a memory write to that address.

CPU Context Switch Time

Table 1 shows the results of six different tests performed to measure the CPU context switch time. Times were measured using two real-time programs, TaskA and TaskB. Immediately prior to the context switch, TaskA assigns a constant to a variable whose physical address is known. The BPM will trigger on this memory write event and begin recording bus activity. Immediately upon gaining control after the context switch, TaskB assigns a constant to a variable and the BPM will stop recording bus activity and display its trigger buffer, including the total elapsed time, to its dedicated terminal. Figure 1 depicts the sample code for TaskA and TaskB used for the H.SURE tests, while Figure 2 reveals the same for the H.SIMS tests. The subroutine, MYWAIT, is an assembler language interface to the H.SIMS module. By inlining the assembly code to H.SURE, the H.SURE tests received a slight advantage.

The first distinction between the six tests is whether the cache memory system was on or off. In the two cache-off tests, the computer was booted after the cache memory boards in both the CPU and IPU buckets were switched off. One cache-off test was performed with the IPU idle while the second test was performed with the IPU executing an infinite loop of 64K consecutive TRR (transfer register-to-register) instructions. The cache-off tests should provide a general

indication of the total number of instructions executed by the two competitors. H.SIMS performed between 80 and 144 microseconds faster than H.SURE, indicating that it used fewer instructions. The most curious result (for which I have no explanation) was that the additional bus activity generated by the IPU actually sped up the H.SIMS execution.

In the four cache-on tests, the first distinction is whether or not TaskA calls a cache purging subroutine before the context switch. Figure 3 depicts the code for the cache purging subroutine. By executing 64K consecutive TRR instructions, the instruction cache should be entirely purged before making the H.SURE or MYWAIT call. Two tests were run with the cache on and TaskA purging cache. In one test, the IPU was idle while in the other test the IPU was executing a task that was continuously purging the cache. Figure 4 depicts the code for the cache purging task.

Depending on the cache addresses given to the context switching code, it is possible that all of the instructions between the start and stop triggers were still in cache in the two tests that were run in which TaskA did not call the cache purging subroutine. The only other code being executed by the computer was the code necessary to resume TaskA, and the trivial amount of code within TaskA and TaskB. Although it would have been possible to do so, it was not confirmed that in these two tests the entire code was indeed cached. However, the BPM records all bus activity during the triggered interval, and the extremely low bus activity during these tests support the conclusion that a large part of the code was cached and, hence, that these times can be used as a best case. If all of the code was cached, then the execution times would again reflect the total number of instructions executed. Here again, we see that H.SIMS appears shorter than H.SURE.

Assuming that real applications would often purge the cache before passing control to the next application, and that the IPU would also be in execution, then the most important times are cache-on, CPU purging, and IPU purging. Here, we see that H.SURE averaged 367 microseconds to H.SIMS 406 microseconds. H.SIMS is designed with absolutely no subroutine calls and cannot, therefore, benefit from cache that is purged. The fact that H.SURE won the two CPU purging tests indicates that it does reexecute some code, allowing it to gain the benefit of the cache.

The MPX documentation on H.SURE says that the priority of the task to be resumed should be higher (lower number) than that of the calling task. The big problem with that is that it requires the user to run all of his tasks in reverse priority order which totally destroys the overrun protection gained from multi-tasking in the first place. H.SIMS runs up to ten real-time tasks in decreasing priority order in the CPU and up to ten tasks similarly in the IPU. If one of the lower priority tasks is still in execution at the top-of-frame interrupt, it will be switched out and the first (highest priority) task will run as it should. In all of the context switch tests performed, TaskA was higher priority (lower number) than TaskB. This resulted in H.SURE scoring considerably longer times than it would have with the priorities reversed. Since H.SIMS is not capable of running tasks in reverse priority order, it was decided that the tests should be equal in this regard.

Lastly, it should be explained how TaskA was resumed. In the H.SURE tests, an interactive terminal was used to run the OPOOM task and resume TaskA. Carriage returns were then used to repeat the command for the one-hundred data points acquired in each test. Clearly, each resumption resulted in a single context switch between TaskA and TaskB which generated a single

triggered buffer on the BPM. In the H.SIMS tests, the built-in top-of-frame interrupt handler was used to resume TaskA at 30Hz. The BPM cannot display its trigger buffer more often than once every two seconds. Hence, the one-hundred data points represent a sample cut of more than 6000 current switches.

IPU Context Switch Time

The method used to measure the IPU context switch time was identical to that employed in the CPU tests described above. The only difference being that the two tasks were catalogued "ipubias". Now we are measuring the time elapsed from the moment TaskA calls the context switcher, from the IPU, until the moment TaskB resumes execution, in the IPU. The BPM made verification, that the origin of both the start and stop events was indeed the IPU, simple. The BPM can distinguish between a memory write originating in the CPU and one originating anywhere else, depending on which of two transaction types the user selects when defining the event trigger. By defining both the start and stop triggers as non-CPU memory writes, and if after resuming TaskA the BPM captures a trigger buffer, then an IPU context switch has occurred.

The MPX documentation on H.SURE says that it should not be used for IPU tasks, while suggesting no alternative. Having studied the source code for H.SURE, and the scheduling modules H.EXEC, H.CPU, and H.IPU, I will state that MPX offers nothing better than H.SURE for context switching between real-time tasks either in the CPU or IPU. Granted the major benefit from H.SURE is realized in the special case of a CPU task resuming a higher priority CPU task, the special design of H.SURE nevertheless insures superior performance over sequential resume and suspend SVC calls, especially for IPU-biased tasks. H.SURE performance for

IFU tasks may be roughly equivalent to suspend SVC calls where an external interrupt handler resumes all IFU tasks at top-of-frame, although this was not investigated. The context switch time was measured with the two IFU tasks in reverse priority (though the results presented are all with standard priority order) and, as expected, there is no statistically significant improvement in the time as there is for CPU tasks.

Table 2 shows the results of the six different tests performed to measure the IFU context switch time. The purpose of H.SIMS was to correct the inadequacies of MPX for real-time IFU context switching. Here we see that in the theoretical best case, that of IFU cached, CPU idle, H.SIMS is 220 microseconds faster than H.SURE. In the theoretical worst case, IFU purging, CPU purging, H.SIMS is 293 microseconds faster than H.SURE. It is interesting to note that H.SIMS is far more immune to the activity of the CPU than is H.SURE. Remember that in order to context switch the IFU, the CPU must intervene. If the CPU is idle, the necessary code to context switch the IFU will remain ready in the CPU cache. Therefore, purging the CPU cache will amplify that portion of the total context switch performed by the CPU.

CPU Overhead Required by IFU Context Switch

Whenever the IFU stops execution of its current task, which it will for a variety of reasons, including most SVC calls, the CPU is trapped in a special trap handler, H.CPU (which is more similar to an interrupt than a trap because it is the only trap that can be blocked by the BEI (block external interrupts) instruction). The H.CPU handler will execute and return via S.EXEC5 which invokes the dispatcher (MPX context switcher) S.EXEC20. If the CPU was currently executing a task when the trap occurred, that task will not continue until S.EXEC20 dispatches back to it.

To know the total overhead required by an IFU context switch, the CPU overhead must be measured as the time elapsed from the trap until the interrupted task regains control of the CPU.

Table 3 shows the results of the five different tests performed to measure the CPU overhead associated with an IFU context switch. In order to measure the CPU overhead, an interactive task executing an infinite loop of alternating STW (store word) instructions was placed in the CPU. In the two CPU-cached tests, the loop was tight with only two STW instructions (see Figure 5). In the remaining three tests, the loop was 64K long (see Figure 6). The BPM was set up to start on a CPU memory write to the first word of the TCB (trap context block) for H.CPU, which is the location where the PSD1 (program status double word one) of the trapped CPU task is stored. The stop trigger was a CPU memory write to the location labeled DATUM in the infinite loop (the target of the STW instructions).

The difficulty in making this measurement lies in the fact that for each frame, you get two CPU traps: the first occurs when TaskA leaves the IFU, requiring the CPU to switch TaskB into the IFU, while the second occurs when TaskB leaves the IFU, requiring the CPU to merely suspend TaskB. The first case is the one involved in a true IFU context switch and, in fact, consumes more CPU overhead than the second case. It was, therefore, necessary to distinguish between the two. Since the BPM records all bus activity between the start and stop triggers, the presence of IFU bus activity in the triggered buffer was used to identify the first, or true IFU context switch event. When the trap occurs, the IFU is idle and not accessing the bus, a condition which will remain true throughout the CPU overhead event if there is no other tasks to be run in the IFU. This is the case when TaskB suspends itself and was confirmed by zero IFU bus

requests, together with shorter event times. When TaskA suspends itself, however, the IPU will begin processing TaskB before the CPU returns control to the trapped task, and this will be reflected by the presence of IPU bus activity as well as longer event times.

considering the difference in the price of the two context switches.

Referring to Table 3, we see that in the theoretical best case, that of IPU cached, CPU cached, H.SIMS is 298 microseconds faster than H.SURE. The fascinating thing is that for both modules, the actual best case was that of IPU purging, CPU cached. I can offer no explanation for this result, but here H.SIMS is 293 microseconds faster than H.SURE. In the theoretical, and actual worst case, that of IPU purging, CPU purging, H.SIMS is 331 microseconds faster than H.SURE.

Total IPU Context Switch Overhead

By computing the total overhead associated with an IPU context switch, we see the truly dramatic advantage of H.SIMS. Taking the theoretical, and most likely, worst case of IPU purging, CPU purging, we see that the total overhead for H.SURE is 1.12 milliseconds, whereas for H.SIMS it is 496 microseconds, a difference of 624 microseconds per IPU context switch. Referring back to Table 1, we see that for H.SIMS, the worst case IPU switch overhead of 496 is only 90 microseconds longer than the worst case CPU switch overhead of 406 microseconds. This means that there is rough parity between the two and, hence, there exists no reason not to multi-task the IPU as heavily as the CPU. The same cannot be said for H.SURE where the difference is 753 microseconds. Using H.SURE with reverse priority order, the difference would be greater still. The irony of MPX is that by providing a means (though unattractive) of making very fast CPU context switches while offering nothing for the IPU, it forces the user to conclude that he cannot afford to multi-task the IPU

Table 1. CPU CONTEXT SWITCH

	CACHE ON				CACHE OFF	
CPU	CACHED		PURGING			
IPU	IDLE	PURGING	IDLE	PURGING	IDLE	BUSY
H.SURE	125	142	313	367	631	658
	2.21	2.35	17.4	20.6	2.63	4.62
H.SIMS	105	115	346	406	551	514
	1.38	1.80	1.02	1.84	3.66	1.89

Table 2. IPU CONTEXT SWITCH

	CACHE ON				CACHE OFF	
IPU	CACHED		PURGING			
CPU	IDLE	PURGING	IDLE	PURGING	IDLE	BUSY
H.SURE	337	499	476	661	1589	1819
	1.52	4.91	2.28	6.48	8.31	5.87
H.SIMS	117	153	306	368	476	492
	1.18	1.50	1.01	1.77	2.38	5.72

Table 3. IPU CONTEXT SWITCH (CPU OVERHEAD)

	CACHE ON				CACHE OFF
IPU	CACHED		PURGING		
CPU	CACHED	PURGING	CACHED	PURGING	BUSY
H.SURE	342 1.55	422 3.93	333 1.74	459 5.14	1775 7.08
H.SIMS	44 0.57	118 0.85	40 0.00	128 2.53	199 1.03

Note: All times are expressed in microseconds. The upper number represents the mean of one hundred data points, while the lower number is the standard deviation of the sample.

<pre> C** TASK_A REAL-TIME LOOP 10 ISTART = 1 INLINE LW 7,NISKNO SVC 5,0 ENDI CCC CALL CACHE GOTO 10 </pre>	<pre> C** TASK_B REAL-TIME LOOP 10 CONTINUE INLINE LI 5,0 TRR 5,6 TRR 6,7 SVC 1,X'54' ENDI ISTOP = 2 GOTO 10 </pre>
--	--

Figure 1. Real-time loop source code for testing H.SURE context switch times. The call to subroutine CACHE was commented out when no cache purge was desired.

<pre> C** TASK_A REAL-TIME LOOP 10 ISTART = 1 CALL MYWAIT CCC CALL CACHE GOTO 10 </pre>	<pre> C** TASK_B REAL-TIME LOOP 10 CALL MYWAIT ISTOP = 2 GOTO 10 </pre>
--	--

Figure 2. Real-time loop source code for testing H.SIMS context switch times. The call to subroutine CACHE was commented out when no cache purge was desired.

```

*
      PROGRAM   CACHE
      DEF       CACHE
*
      LIST     NOREP
      M.EQUS
*
      CACHE    EQU      $
              REPT     32767      32K BYTES
              TRR      R2,R3      SILLY INSTRUCTION
              ENDR
              REPT     32767      32K BYTES
              TRR      R2,R3      SILLY INSTRUCTION
              ENDR
              TRSW     R0          RETURN
*
      END
*

```

Figure 3. Instruction cache purge subroutine source code.

```

*
      PROGRAM   PURGE
      DEF       PURGE
*
      LIST     NOREP
      M.EQUS
*
      PURGE    EQU      $
              REPT     32767      32K BYTES
              TRR      R2,R3      SILLY INSTRUCTION
              ENDR
              REPT     32767      32K BYTES
              TRR      R2,R3      SILLY INSTRUCTION
              ENDR
              BU        PURGE      GO FOREVER
*
      END
*

```

Figure 4. Instruction cache purge task source code.

```

*
*      PROGRAM  LOOP
*      DEF      LOOP
*
*      LIST     NOREP
*      M.EQUS
*
*      MPX SYSTEM EQUATES
*
*      DATUM    DATAW  Ø      TARGET
*
*      LOOP     EQU      $
*              LI      R4,1
*              LI      R6,2
*
*      CONTIN   EQU      $
*              STW     R4,DATUM    WRITE 1
*              STW     R6,DATUM    WRITE 2
*              BU      CONTIN      GO FOREVER
*
*      END
*

```

Figure 5. Task source code with continuous memory writes.

```

*
*      PROGRAM  LOOP
*      DEF      LOOP
*
*      LIST     NOREP
*      M.EQUS
*
*      MPX SYSTEM EQUATES
*
*      DATUM    DATAW  Ø      TARGET
*
*      LOOP     EQU      $
*              LI      R4,1
*              LI      R6,2
*
*      CONTIN   EQU      $
*              REPT    16384      64K BYTES
*              STW     R4,DATUM    WRITE 1
*              STW     R6,DATUM    WRITE 2
*              ENDR
*              BU      CONTIN      GO FOREVER
*
*      END
*

```

Figure 6. Instruction cache purge task source code, with continuous memory writes.

Brian F. Goldiez and Kevin C. Uliano
 Institute for Simulation and Training
 University of Central Florida
 and
 Dennis McBride
 Defense Advanced Research Projects Agency

Abstract

This paper describes efforts to develop a low cost aviation technology testbed to investigate the flying qualities and performance parameters of limited fidelity flight simulators. Initial efforts in this testbed have revolved around specifications and preliminary feasibility testing of a video-based data acquisition system. Initial pilot-in-the-loop experiments have also been conducted to investigate the usefulness of pilot opinion as data. The preliminary results from this testbed are encouraging. Further efforts will center on the identification and testing of quantitative measures of simulator fidelity.

Introduction

Development of low cost simulations that can be produced in large numbers is of particular concern to the Department of Defense (DoD) (U.S. Department of Defense, Defense Science Board, 1982). More recently, the DoD has formulated an investment strategy plan for critical technologies. The areas of simulation and modeling, ranked fifth, will "... play an increasingly important role in the cost effective training of personnel, as well as in the design of equipment that can be operated and maintained efficiently." (U.S. Department of Defense, 1990, p. A-55).

Historically, simulators -- especially flight simulators -- have been developed with a single user in mind. Recent and continuing advancements in computer technology, visual presentation techniques, and real time data communication capabilities are beginning to allow simulators from different manufacturers that differ in their level of realism to be linked or networked together. In 1989, the first conference on Interactive Networked Simulation for Training was held to provide a forum for the identification of user networking requirements and the related research areas that must be investigated to resolve behavioral, engineering, and training questions surrounding simulation in a networked environment (Gilson, Kincaid, and Goldiez, 1989).

Explosive leaps in technology for simulation must be tempered by our lack of understanding of the impact of simulation technology on the human. While the goal is to extend the comprehensiveness of simulation while reducing costs, questions arise which must be addressed. The National Research Council's Working Group on Simulation (Jones, Hennessy, and Deutsch, 1985) outlined three recommendations in an attempt to fill in knowledge gaps on simulation for training:

1. Long-range, comprehensive, and forward-looking research plans should be developed to address persistent and emerging simulation problems.
2. Long-range stable funding should be provided to encourage the development of academic bases for simulation research.
3. Research to develop near-real-time human performance assessment capability for simulation is urgently needed.

One recurrent problem is the construct of simulator fidelity.

The term "fidelity" is unclear because it has been used in a wide variety of contexts to imply a wide variety of meanings. (see Hays and Singer, 1989, for a review of the simulation fidelity literature). For example, there is equipment fidelity, physical fidelity, and behavioral fidelity. Moreover, these labels mean different things to different people. From a training point of view, the question becomes how to represent reality within the training situation so that established training criteria are met in the most efficient manner. From an engineering point of view, the question centers on the ability of the simulator to represent the physical performance characteristics of the device it represents.

The Institute for Simulation and Training (IST) at the University of Central Florida is presently under contract to the Defense Advanced Research Projects Agency (DARPA) and the Army's Project Manager for Training Devices (PM TRADE) to investigate fidelity and networking issues in low cost aviation simulation. As part of this program, IST has procured two commercially available F-16A simulators. These devices, known as the Avionics Situational Awareness Trainers (ASAT), are MS-DOS based part task simulators developed ostensibly to train beyond visual range (BVR) target acquisition and engagement, both for an individual aircraft and in a networked environment. ASAT is described in greater detail later in this paper.

Problem

During the procurement process which culminated in the purchase of ASAT, it became apparent that flight and acceptance testing of such devices to determine performance and handling qualities was haphazard at best. Secondly, government procurement agencies did not appear to know what to specify in such devices with respect to handling qualities and performance to meet a particular simulation objective. In general, simulator procurement specialists and users share an understanding of hardware technology, but usually neither group is well versed in training, human performance measurement, or human factors engineering. All too often the design, procurement, and employment of simulators are separate activities with little or no coordination between parties.

This initial research effort involves development of initial flight performance tests to answer two questions: (1) How does objective and subjective ASAT flight performance in a single axis maneuver compare with the same performance in the aircraft? (2) What factors influence the handling qualities and performance of ASAT with a pilot-in-the-loop?

Research Methodology

Apparatus

The basic ASAT is presented in Figure 1.



Figure 1. Avionics Situational Awareness Trainer (ASAT)

The ASAT F-16A part task simulator was designed and manufactured by Perceptronics, Inc. to provide training to pilots in the beyond visual range (BVR) environment. Throttle and side stick controller are faithful reproductions. The Radar Warning Receiver (RWR), Radar Electro-Optical Display (REO), and Head-Up Display (HUD) are also accurate representations of the actual displays in the aircraft with generic symbology substituted for classified symbology. The HUD is overlaid on the single out-the-window-display. Field of view is 23 degrees vertical by 23 degrees horizontal. Weapon and radar audio tones, limited vibrational cuing and countermeasures (i.e., AIM 9-L/P missiles, flares and chaff) are also represented.

Data Acquisition System Design

Flight testing of the aircraft typically involves the production of control inputs and capture of output information which contain physical variables. Data capture systems normally use gyroscopes and accelerometers to collect analog information. Noise or error is often introduced into the system because of vibration, turbulence, and RF interference. Stable control conditions are also difficult to duplicate consistently. Therefore, flight test data are typically subject to considerable post processing to obtain useful information.

Most limited fidelity flight simulators such as ASAT rely heavily on visual representation and presentation of the external environment. Tactile and audio information are usually secondary cuing system in such devices. Therefore, it becomes necessary to capture simulator flight data in a form that can be analyzed against actual aircraft performance data. Data from the simulator must also preserve the temporal relationships between data points.

A suite of test equipment is being designed by IST to capture flight simulator performance data. Ideally, such equipment should gather data without intruding into the simulator or affecting its operation. In addition, sampling rates should be such that critical data are not lost and time skewing is minimized. Figure 2 shows a diagram of the proposed test equipment design. This design is based on an existing configuration currently being used at the Naval Air Test Center (NATC) during actual flight tests. The design criteria center around capturing data in an analog format, so that data can be gathered from standard output devices. Videotape is the initial medium. This approach avoids intrusion into the system and the problems associated with discrete sampling back to the baseline system. Problems associated with proprietary hardware and software are also avoided.

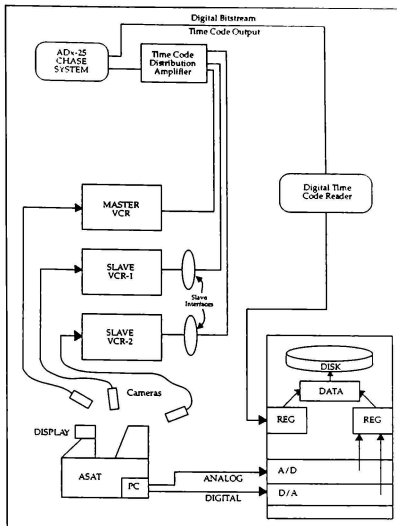


Figure 2. Proposed data acquisition system design.

Analog-to-digital and digital-to-digital converters are also being used to capture control information not amenable to videotape. In the F-16A this information consists of control stick inputs (as the control stick is stationary and is responsive to pilot force inputs). In addition several discrete controls on the stick and throttle such as missile cage/uncage, missile select and radar mode select are necessary information for both human factors analyses and flight performance tests.

Sampling rates are critical in flight test methodology. The suitability of using videotape as a data recording medium was analyzed. Review of Nyquist's sampling theory dictates that the sampling frequency should be at a rate at least twice the natural frequency of the system under study (Haykin, 1978). Typical values of the natural frequency of high performance jet aircraft is 3 Hz (Roskam, 1972). Videotape is limited to 60 Hz field rates for compatibility with NTSC television standards. This rate is an order of magnitude greater than the nominal frequency of an aircraft. Video sampling therefore represents a viable method to gather flight data.

Although only three videocameras will be used in the data acquisition system portrayed in Figure 2, up to four cameras can be accommodated. One camera will be used for the HUD, one for radar displays, and one for stick and throttle. Capture of analog information is controllable at nominal frequencies of up to 4000 Hz for all analog signals. This is well within the range of the individual aircraft's natural frequency.

There are certain problems associated with video capture techniques and the parameter identification method. Parameter identification is an approach to glean parameters from aircraft flight tests that can be used for simulation modeling purposes. Unfortunately, these methods are subject to noise from many sources. It is important to remember that the goal of this research is to validate flight simulator performance. As such, when data are

pulled from instruments, indicators, or HUDs, performance parameters may become corrupted in ways that differ from the aircraft due to phenomena such as aeroelastic bending and vibration. Therefore, the relationship between the simulation and aircraft from an engineering point of view is not intuitively obvious.

The parameter identification method will not only be a valid method to compare simulator performance to the actual aircraft, but will also be useful in cue replacement analysis. Collecting information from flight instruments is being conducted in a manner similar to that which the human uses to receive information -- that is visually; therefore, the primary source of stimuli for the trainee to process is visual in nature. These stimuli have to serve as the primary source of information in an otherwise cue-depleted environment. For example, the cues experienced by a pilot while executing a simple aileron roll are visual, vestibular, and tactile. Most limited fidelity devices do not have control loading systems to provide tactile feedback, nor do they contain acceleration systems or onset cuing to provide vestibular information. The addition of low cost control loading systems coupled with a quality sound and vibrational system for limited kinematic cues may be necessary to increase the usefulness and acceptance of many limited fidelity devices.

Data Acquisition System Feasibility Testing

Flight tests were performed to test the feasibility of the proposed Data Acquisition System discussed above. Flight test data were requested and obtained from the U.S. Air Force F-16 Program Office. Roll performance tests were replicated in the ASAT. Performance tests include time to achieve a 30, 90, 180, and 360 roll attitude, respectively, at a stable mach number, altitude and a 1-G flight condition. The intent was to keep the flight performance confined to a single axis for simplicity purposes. The authors served as test pilots for this initial study. These maneuvers were filmed on Sony CCDF-35 8mm videotape via the test setup in Figure 3. The source tape was then dubbed to a Sony VO-9850 videotape editor for frame-by-frame analysis. There were several purposes for these initial flight tests. The first purpose of the flight tests was to investigate any video synchronization problems between the HUD and the video camera. The second objective was to note any physical restrictions in data gathering such as camera setup, lighting, and special fixtures. The final purpose was to determine if sufficient resolution and contrast existing in the source videotape.



Figure 3. Feasibility testing design of proposed data acquisition system.

Video synchronization differences are apparent in the tests. A retrace band is often visible due to differences in the synchronization in the vertical retrace of the camera and source imagery. This effect is quite common and can be alleviated by using an external synchronization signal to control the image source

and image collection device. Close inspection of the videotape reveals an apparent loss of contrast but not resolution. Therefore, if high contrast ratio image source is used, videotape should be quite acceptable.

Physical restrictions to gathering data with this setup were also investigated. Lighting effectiveness proved to be the variable with the greatest degree of sensitivity. First, some aircraft instruments are sources of light (i.e., self-illuminating), while other instruments require an external lighting source. For instance, in the initial experiments discussed here the HUD was a light source while a LCD stopwatch required external lighting to be visible to the camera. External lighting caused glare on the CRT face containing the HUD and out-the-window display. The amount of glare was a function of the orientation of the light source. Therefore, a focused light source was aimed at the stopwatch to provide sufficient lighting to view both images from a single unmodified aperture. Other noted constraints are indicative of the simulation configuration. A standard camera tripod and commercial test fixtures can accommodate most configurations on limited fidelity devices. Camera size and weight, however, should be minimized to lessen visual obstruction caused by the camera or its supporting structure.

Remote videotaping is recommended and will be investigated on subsequent tests of this system. Additional manned flight testing will also be performed using the operational simulation validation hardware, and expanded flight tests to facilitate parameter identification studies. Tests will include both fixed and rotary wing aircraft and simulators.

Manned Flight Testing

There were two purposes of the manned flight testing using ASAT. The main objective was to replicate the roll performance tests discussed previously; this time with qualified F-16 pilots. Since it was not feasible to videotape these maneuvers due to the awkwardness of the test setup (see Figure 3), subjective aircraft handling and performance data were collected and compared to the flight performance data produced from the analysis of the videotape. The secondary objective was to conduct an initial empirical investigation into the tasks a pilot performs while flying realistic scenarios in a limited fidelity simulator. These task components were studied to determine what cues are available in the simulator versus what cues are useful in task performance.

Subjects. Two F-16 qualified pilots served as subjects for this experiment. They were from the 56th Tactical Training Wing at MacDill Air Force Base, FL. They reported 1100 and 750 flight hours, respectively in the F-16A.

Flight Scenarios. The pilots flew a five minute familiarization flight followed by five scenarios. Two scenarios were single ship in which each pilot flew separately against one or two aggressors. Three scenarios were in the team mode where the two pilots flew in team formation against multiple computer-generated aggressors. The aggressors appeared as either MiG-21 or MiG-29 aircraft. Initial flight conditions of both the friendly (blue) and hostile (red) force are presented in Table 1.

Table 1. Initial flight conditions.

		Scenario				
		1	2	3	4	5
BLUE FORCE	Altitude	25,000ft.	25,000ft.	24,000ft.	18,500ft.	30,000ft.
	Heading	315°	0°	0°	0°	0°
	Single Ship(S) or Team (T)	S	S	T	T	T
RED FORCE	Altitude	20,000ft.	31,000ft.	15,000ft.	20,000ft.	20,000ft.
	Heading	220°	180°	180°	180°	180°
	Distance	42mi	20mi	30mi	30mi	28mi
	Number of Aggressors	1	2	2	3	4

ACQ-1007

Cooper-Harper Aircraft Handling Characteristics Scale

Subjective pilot rating is still the common method of assessing the handling qualities of an aircraft and in determining its suitability for an intended mission (Ellis, 1978). The most widely used rating procedure is the Cooper-Harper Aircraft Handling Characteristics Scale (Cooper and Harper, 1969) originally designed for use by test pilots. The scale shown in Figure 4 deals primarily with aircraft ease of control with numerous references made to task demands and pilot compensation. The pilot is guided through by a step-by-step rating process. The value judgements that the pilot makes are presented as a series of decisions. The dichotomous choices at the first two stages (i.e., from left to right) of the decision "tree" are fairly simple. From there the pilot makes a decision from three final choices. The scale is from 1 to 10 with "1" being excellent or highly desirable to 10 being totally uncontrollable. The original Cooper-Harper scale and its derivatives have been used for over 20 years and have been extensively researched and validated.

Although the Cooper-Harper deals mainly with aircraft ease of control, O'Donnell and Eggemeier (1986) provide some evidence in support of using the scale as a workload index. Hess (1987) has also provided data supporting a monotonic relationship between the Cooper-Harper scale and operator load. The interpretation of the Cooper-Harper

ratings in this study, however, was limited strictly to simulator handling and performance qualities.

The test pilots in this study were requested to complete the scale at the conclusion of each scenario and at the end of each of the individual roll maneuvers. Both pilots were instructed on the purpose and use of the Cooper-Harper Scale.

Verbal Protocol Analysis

Part of identifying the design parameters of limited fidelity flight simulation devices is to identify the cues that are used by pilots in performance of their tasks. Once these cues are identified then, to some extent, they must be modeled in the simulation. Verbal protocol analysis (Ericsson and Simon, 1984) is one methodology for analyzing tasks into behavioral components. The protocol is an audiotaped record and transcription of the pilot's verbalizations during the course of simulated flight scenarios and thus provides information in a real-time, dynamic environment. It is particularly useful in analyzing phenomena that exhibit strong historical dependence. Verbal protocol analysis is also a very useful intermediate step between full specification of all variables which may take a long time to assemble, and the alternative of specification of physical reality which implies we do not know how to abstract the task.

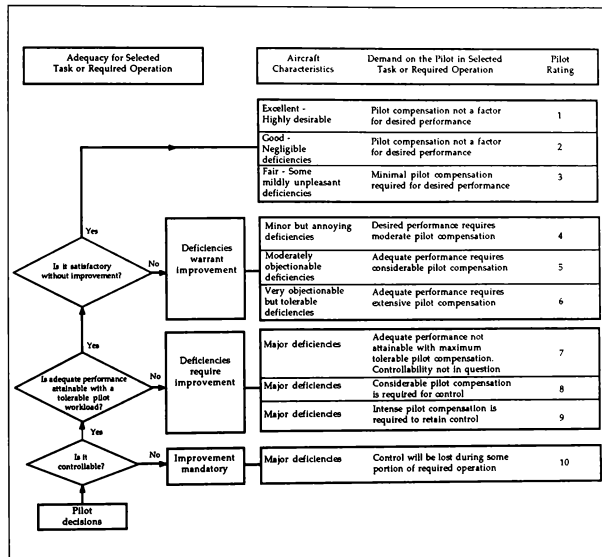


Figure 4. The Cooper-Harper aircraft handling characteristics scale.

In Ericsson and Simon's work, the protocol of the subject is analyzed to represent what the subject knows and what perceptual, cognitive, and response operations are being applied. The experimenter infers what the subject knows from his verbalizations according to a systematic and formal set of procedures, together with the experimenter's knowledge of language and his/her ability to extract meaning. The goal of the protocol analysis in this current study was simply to develop insights into the behavior and task components associated with pre-merge and merge air-to-air engagements. Berbaum and Kennedy (1985) used this procedure to identify the stages of visual information processing during a simulated helicopter shipboard landing.

Procedure

Pilots were first shown a brief videotape introduction to the ASAT and its systems. Next, they were given printed briefing materials which outlined the research objectives, the controls and displays of the ASAT, the nature of the flight scenarios they would be flying, and the purpose and use of the Cooper-Harper scales. They then flew a five minute familiarization flight. During this trial flight, they practiced "thinking aloud". That is, as part of the verbal protocol analysis discussed above, it was necessary to record their verbal responses to those aspects of the scenario they were paying attention to and the behaviors they were trying to execute. This verbal information was recorded via the ASAT headset into separate audio tape recorders for later analysis. Since the pilots were also instructor-pilots and were accustomed to explaining aircraft and systems operations to student pilots, this was not an unrealistic request.

Each pilot then flew the first two flight scenarios in a single ship mode. The remaining three scenarios were flown in team mode. Audio traffic was monitored to ensure that the pilots kept verbalizing what they were doing; reminders were given as necessary. Each scenario was flown in order and was repeated twice. After each scenario was completed the pilots completed Cooper Harper ratings of the simulation based on the given flight scenario presented.

After all flight scenarios were flown, the pilots replicated the roll performance tests discussed earlier. Cooper-Harper ratings strictly on the roll performance of the simulator were also collected. Finally, the pilot completed a debrief questionnaire and rating form on both general and specific aspects of the simulation.

Results

Roll Performance

The roll performance data were analyzed to investigate two critical simulation issues. First, was there sufficient correlation between the actual F-16A roll performance data and the data produced from the ASAT roll performance experiments. Second, to what extent did the pilots' subjective rating of the handling qualities of the ASAT, while performing the exact same roll maneuvers, match the objective data from the experiments. Because the actual F-16A roll flight test data were independent of roll direction, the experimental data were averaged to obtain a mean time for response for each velocity.

Figure 5 shows differences in time to bank 30 degrees (in seconds) between the actual F-16A and the ASAT across the four velocity test conditions along the abscissa. The ideal modeling condition would produce no appreciable difference between the simulator and the aircraft fit models. Results indicated, however, that there were rather large differences (.5-.6 seconds) for the slower velocity conditions. Performance improved at the two highest velocity conditions (250, 300 kts, respectively). Visual inspection reveals a strong decreasing linear component to the data

and, at higher airspeed than were tested, the trend would probably continue.

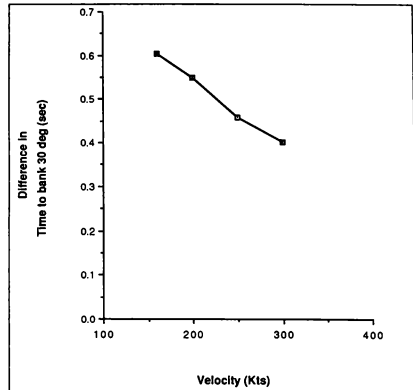


Figure 5. Roll performance comparisons: F-16A versus ASAT flight data.

The F-16 test pilots replicated the same roll experiments in the ASAT. Cooper-Harper ratings were collected for each roll conditions. Results are presented in Figure 6.

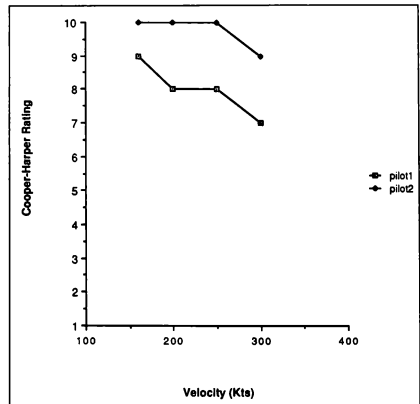


Figure 6. Cooper-Harper ratings as a function of test velocity.

The overall roll performance of the ASAT, in the opinion of the two test pilots, was unacceptable (mean Cooper-Harper rating of 9); however, as the velocity at which the roll maneuver was performed increased, the pilots reported that the simulator handled better. These results follow the same trend seen in the comparison of the flight performance data.

Finally, another graph was drawn showing the pilots' Cooper-Harper ratings plotted against the difference between the

ASAT and F-16A roll performances for each velocity test condition. This comparison is shown in Figure 7.

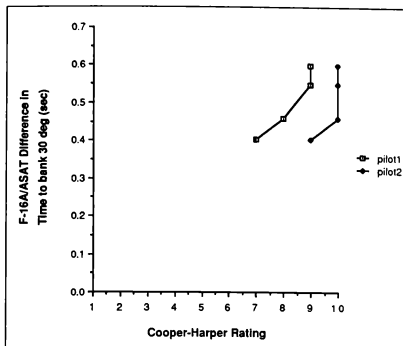


Figure 7. Roll performance comparisons: difference values as a function of pilot opinion.

Data from this chart provides initial evidence to support a monotonic relationship between performance differences in a limited single axis maneuver and experienced pilots' rating on the handling quality of the simulator related to that maneuver. The experimental pilots who served as subjects for this experiment were apparently sensitive to simulator flight performance differences from the reference aircraft, and the Cooper-Harper rating scale was sensitive enough to record those differences.

Protocol Analysis

The verbal protocol recorded from both pilots was analyzed to provide preliminary information on the cues used versus the cues actually available during air-to-air engagement flight scenarios. Other observations and comments regarding the operation and behavior of the simulator were noted. The intent of this analysis was to develop a list of observations and insights.

Approximately 3.5 hours of audiotapes were transcribed. Almost immediately it became apparent that critical aircraft systems such as radar/antenna performance and flight dynamics were not performing in manner similar to the reference aircraft. Most of these problems are inherent in the

software modeling used for ASAT.

The transcriptions were then reviewed to determine a top level classification scheme. This analysis produced five logical classification dimensions within which to place verbal data documenting deficiencies: physical layout/appearance, discrete operations of the systems (switches, discrete controls), continuous operations of the system (dynamics, control response/feedback), characteristics of threats. A total of 65 useful protocols were identified. Table 2 provides summary information of the numbers and percentage of protocols falling in each category listed along with sample protocol excerpts.

The most numerous and serious deficiencies in the simulation centered around the continuous operations of the system including noticeably inaccurate vehicle dynamics. The impact of these deficiencies were lessened by having the pilots calibrate the stick, throttle, radar cursor/enable, and antenna. Also, some discrete controls of the system did not function like the aircraft. In terms of some of the visual and aural cues that were presented, There were incomplete and/or inaccurate representations. Finally, the deficiencies noted in the physical appearance and layout of the simulator were minor.

Table 2. Verbal protocol analysis results and sample excerpts.

Category	# of protocols in category	# of total protocols
Physical appearance/layout	10	15.38%
<u>Sample Excerpt:</u> "...also the uh, this is one switch versus two(chaff and flares). It's just one little button about this big. That's it. You have to select over here if you want chaff, flare, chaff and flare, either, or..."		
Discrete Control	13	20.0%
<u>Sample Excerpt:</u> "... I go back to center position, and it stays in dogfight; until I go to missile override and then come back to center position...and it's not supposed to do that."		
Continuous Control	26	40.0%
<u>Sample Excerpts:</u> "...and again the flight path marker goes out of view when you pull any G's on the aircraft. That's not realistic. It's not as sensitive...the aircraft is less sensitive." "...the dogfight programmer, see I'm only pulling 3 G's. I could never shoot anybody, 4 G's it's off the ...yeah, in order to shoot anybody with a gun you have to be at 1 G. See what it's doing there at 1 G. It's jumping around. I could never get a tracking shot with that, at best it would be slashing..."		
Sensory Simulation	10	15.38%
<u>Sample Excerpt:</u> "We did not get the audio cue."		
Characteristics of Threats	6	9.24%
<u>Sample Excerpt:</u> "They're coming out of missile range. Boy, there's no way they're going that fast. I got 400 kts and they've got 300 kts of going away speed. They don't go that fast. It needs to be more consistent."		

Another interesting observation was that even at greater than 25 miles from merge -- beyond visual range -- the pilots were still looking for visual offset cues provided by their wingman for use in tactical deployment. These cues were not available in the limited field of view provided by the ASAT. Both pilots commented that ASAT would provide good BVR skills training (assuming the problems noted were fixed), but at less than 10 miles from merge, the lack of sufficient horizontal and vertical field of view was a serious problem. Also the realism in both appearance and function of the radar warning receiver and radar electro-optical

displays is critical, especially in a BVR environment. Weapon and radar tones and their frequencies must be precisely modeled since pilots rely heavily on aural feedback. Finally, during networked team mode operations, there was much verbal information being passed between pilots concerning tactics, vehicle status, and aggressor status, indicating that they soon became immersed in the simulation.

Other Analyses

Debrief questionnaire and rating sheets were administered to the pilots to get an overall impression of the performance of specific aircraft/simulator systems. Table 3 lists the dimensions measured along with the mean rating.

Table 3. Debrief rating results.

Dimension	MEAN PILOT RATING				
	1	2	3	4	5
	Unrealistic		Moderately Realistic		Performed exactly like the aircraft
Realism of the flight scenarios					X
Overall flight performance of the Simulator		X			X
"Feel" of the stick controller				X	
"Feel" of the throttle			X		
Performance of the antenna			X		
Performance of the radar			X	X	
Performance of the missiles		X			
Realism of aggressor tactics					X

0603-1015

The realism of the flight scenarios and aggressor tactics received the highest ratings. The overall performance of the simulator received a mean rating of 2.5 which was probably a result of the poor performance of the antenna, radar, and missiles -- the three critical components to successful BVR flying. There was good agreement on the ratings provided by each pilot with the values differing by no more than one-half point. These global ratings agreed nicely with the specific information gleaned from the pilot's verbal information.

Discussion

The comparison of the roll performance of the F-16A versus the ASAT revealed some interesting differences especially at slower velocities. When pilots were asked to perform similar rolls in the simulator, their ratings on the ASAT's handling qualities during the roll maneuver were more favorable at higher velocities indicating general agreement between the objective measures collected via videotape and pilot opinion. More importantly, the pilots were able to detect small deviations (on the order of 100 msec.) in simulator performance compared to the reference aircraft.

This finding reinforces the widely held theory that humans are generally more sensitive to angular accelerations in the roll axis than in either pitch or yaw axes due to the structure and function of the vestibular system.

The reader will note that these are preliminary conclusions drawn partly from the inspection of data from two pilots. More robust studies are planned to further define implications for simulation. Under broader testing conditions it is hypothesized that Cooper-Harper ratings can be reliably used to measure the degree to which a simulator matches the design basis aircraft from a pilot-in-the-loop perspective. Such an approach provides a method to anchor the construct of simulator fidelity. The key to this approach would involve developing a methodology to quantitatively relate the Cooper-Harper rating system to aircraft system performance.

Such an approach is being developed in conjunction with the Naval Air Test Center (NATC). We will begin to evaluate measurement systems to compare the Cooper-Harper scale with system performance. One method under consideration involves using Kalman filtering techniques of stick activity as an index of pilot workload. Another approach would be to use fast Fourier transforms of stick activity to create a weighted task complexity index. Either of these predictor indices could then be statistically related back to the Cooper-Harper Scale as a criterion measure. The next task would involve making selected changes in dynamics, aerodynamics, and other fidelity areas and evaluating how those changes would impact Cooper-Harper ratings. In this way, systematic changes in fidelity can be evaluated against the impact on pilot workload and acceptance.

Acknowledgements

The authors wish to thank Boris Nguyen, Mike Smith, Brad Geidd, Karen Danisas, and Jada Brooks for help in developing the test configuration, data analysis, and report preparation. Our appreciation is also tendered to Dr. Norm Lane for earlier review and comments on this paper.

References

- Berbaum, K.S., & Kennedy, R.S. (1985). *An analysis of visual tasks in helicopter shipboard landing* (Report No. NASA NAVTRAEQUIPCEN 81-C-0105-19). Westlake Village, CA: Canyon Research Group, Inc.
- Cooper G. E., & Harper, R.P., Jr. (1969). *The use of pilot rating in the education of aircraft handling qualities* (Report No. NASA TN-D-5153). Moffett Field, CA: Ames Research Center, National Aeronautics and Space Administration.
- Ellis, G.A. (1978). Subjective assessment pilot opinion measures. *Assessing Pilot Workload* (AGARDograph No. 233). Neuilly Sur, France: Advisory Group for Aerospace Research and Development.
- Ericsson, K.A., & Simon, H.A. (1984). *Protocol analysis: Verbal reports as data*. Cambridge, MA: The MIT Press.
- Gilson, R., Kincaid, J.P., & Goldiez, B. (1989). Interactive networked simulation for training (Foreword). In R. Gilson, J.P. Kincaid, & B. Goldiez (Eds.), *Proceedings of the Interactive Networked Simulation for Training Conference* (UCF-IST-TR-89-2). Orlando, FL: University of Central Florida, Institute for Simulation and Training.
- Haykin, S. (1978). *Communication Systems*. New York: Wiley.
- Hays, R.T., & Singer, M.J. (1989). *Simulation fidelity in training systems design: Bridging the gap between reality and training*. New York: Springer-Verlag.

- Hess, R.A. (1977). Prediction of pilot opinion ratings using an optimal pilot model. Human Factors, 19, 459-476.
- Jones, E.R., & Hennessy R.T., & Deutsch, S. (Eds.). (1985). Human factors aspects of simulation. Washington, DC: National Academy Press.
- O'Donnell, R.D., & Eggemeier, F.T. (1986). Workload assessment methodology. In R. Boff, L. Kaufman, & J.P. Thomas (Eds.). Handbook of perception and human performance (Volume II: Cognitive processes and performance) (pp. 42-1 - 42-49). New York: Wiley.
- Roskam, J. (1972). Flight dynamics of rigid and elastic airplanes. Lawrence, KS: University of Kansas.
- U.S. Department of Defense. (March, 1990). Critical technologies plan for the Committee on Armed Services United States Congress. Washington, DC: U.S. Department of Defense.
- U.S. Department of Defense, Defense Science Board. (1982). Summer study on training and training technology. Washington, DC: Office of the Undersecretary of Defense Research and Engineering, U.S. Department of Defense.

EVOLUTION OF A REAL-TIME AIR COMBAT ENVIRONMENT

Ronald G. Moore*
 Lee P. Emerson*
 Boeing Military Airplanes
 Seattle, Washington

Abstract

Boeing Military Airplanes (BMA) has, as has many other companies, developed a real-time air combat environment simulation. It follows the design philosophy established in the construction of BMA's Integrated Technology Development Laboratories (ITDL). The simulation accommodates multiple participants, both manned and unmanned, in a distributed, real-time, air combat environment simulation. This paper examines the evolution of BMA's multiple participant air combat environment simulation from its beginning in a single vehicle simulation. The paper presents the design of the ITDL's first generation implementation of a multiple participant simulation, and summarizes the lessons learned from the first generation simulation that have contributed to the second generation simulation.

Introduction

There is a trend in the defense simulation industry today toward building large-scale combat simulations. The Defense Advanced Research Projects Agency's (DARPA) sponsorship of SIMNET and AIRNET[1], the U.S. Army's support of MULTISIM[2], the U.S. Air Force's funding of Multi-ship training[3], and the industry/inter-service effort underway to define Standards for the Interoperability of Defense Simulations[4] indicates the Department of Defense's desire to create large-scale combat simulations. Over the last decade many companies in the defense flight simulation industry have turned their focus from single vehicle simulations to large-scale combat simulations. To support these large-scale combat simulations, companies are connecting their single vehicle simulators together, developing low-cost pilot stations, and incorporating sophisticated unmanned computer controlled vehicle simulations[3,5,6].

Boeing Military Airplanes (BMA) has recognized the need to develop and test advanced aircraft technology in air combat missions requiring cooperative operations. With potentially large force-on-force ratios[7] facing current and next generation military vehicles, advanced aircraft technology supporting cooperative operations has become essential to mission effectiveness. BMA

has developed a distributed, real-time, multiple participant air combat environment simulation, that supports the research and development of advanced aircraft technology.

This paper will present BMA's evolution of a multiple participant air combat environment simulation. The development of BMA's multiple participant simulations has been influenced by the expansion of the Flight Simulation organization and the construction of the Integrated Technology Development Laboratories (ITDL). The development of the multiple participant environment simulation began with BMA's single vehicle simulation. An analysis of the strengths and weaknesses of BMA's single vehicle simulation, with respect to air combat, led to the design requirements for a multiple participant simulation. The first generation implementation of a multiple participant simulation is referred to as HIFEN1. The second generation simulation, referred to as HIFEN2, benefited from the lessons learned in the implementation of HIFEN1. HIFEN1 and HIFEN2 are the ITDL's tools for the integration and evaluation of advanced aircraft technologies.

Background

BMA's Flight Simulation organization was established in 1964** to provide flight simulation support to Boeing engineering research and development activities. The Flight Simulation organization supports the

**The Flight Simulation organization was originally established under the Boeing Aerospace Company (BAC). Later, the Flight Simulation organization was placed under the Boeing Military Airplane Division (BMAD) of the Boeing Aerospace Company. Not long after its placement under BMD, BMD was broken-off from BAC, and was reorganized to form the Boeing Military Airplane Company (BMAC). Since the construction of the Integrated Technology Development Laboratories (ITDL) the Flight Simulation organization has become known as the Integrated Technology Development Laboratories Staff, and Boeing Military Airplane Company has been reorganized and renamed the Boeing Military Airplanes (BMA) division of the Boeing Defense and Space Group (BD&SG).

*Member AIAA

evaluation of flight control systems, aerodynamic models, crewstation designs, spacecraft control systems, avionics systems, and other flight-related research efforts. On occasion, the flight simulation capability serves as flight and procedures trainers for company and customer pilots. In the late 1970s the Flight Simulation organization occupied three laboratories isolated from BMA's other military aircraft research and development laboratories.

In the early 1980s, BMA began construction of a new facility to consolidate their military aircraft research and development laboratories. BMA could no longer afford to develop aircraft systems in isolation from one another; the complexity of developing new military aircraft had outgrown conventional methods. A more integrated approach to military aircraft technology development was required.

By November of 1986 the new Integrated Technology Development Laboratories (ITDL) facility was complete. The ITDL is a tempest facility housing twenty-two independent, securable laboratories. This facility houses three thirty foot domes, a motion base, four host computer rooms, two non-visual low-cost cockpit laboratories, two Computer Image Generator (CIG) laboratories, a CIG data base development laboratory, and a number of military aircraft technology and project dedicated laboratories. By having independent laboratories, the ITDL allows each laboratory to operate autonomously or in conjunction with other laboratories. Laboratories operating in conjunction with one another communicate through reconfigurable fiber-optic networks[8].

The construction of the ITDL consolidated BMA's military aircraft research and development laboratories, and gave the Flight Simulation organization the opportunity to develop a new, more generalized simulation architecture. This development began with BMA's single vehicle simulation.

Single Vehicle Simulation

The basic components of the single vehicle simulation architecture have been used by BMA's Flight Simulation organization since 1964. The basic components included a host computer (executing the vehicle simulation), a cockpit in front of a projection screen, and the hardware for in-cockpit and out-of-cockpit imagery.

With regard to modern air combat, the single vehicle simulation has both strengths and limitations. The single vehicle simulation focuses on the tasks of a single crew. It serves well for part-task studies, such as flight controls development, pilot-vehicle-interface definition, and handling qualities evaluations. However,

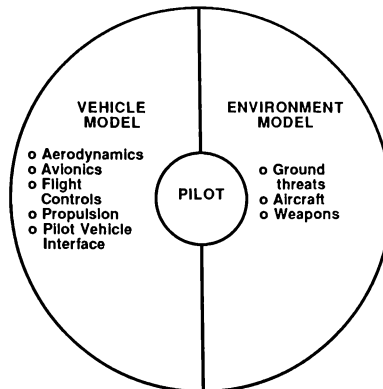


Figure 1. The Single Vehicle Simulation

because of limitations in computational resources, the typical single vehicle simulation is limited in its ability to provide a target rich environment representative of modern air combat.

The perspective of the single vehicle simulation is one crew, one task, and one mission. As conceptually shown in Figure 1. (The Single Vehicle Simulation), the pilot/crew is the focus of the single vehicle simulation. The vehicle model provides the crew with the illusion of a real aircraft and the environment model provides the crew with the illusion of a real world. For some studies, the environment model contains simple "canned" aircraft. Other studies require a more robust environment model containing ground threats, computer controlled aggressor aircraft, weapons, and other combat elements. In the single vehicle simulation, combat elements in the environment model are limited to interactions which supports the illusion being created for the crew. Combat elements in the environment model interact only with the vehicle model.

The single vehicle simulation performs well for studies of the vehicle model and its sub-systems. The number of control variables can be limited, allowing deterministic evaluations of vehicle sub-systems. The environment model in a single vehicle simulation allows repeatability and determinism of the combat elements. Typically, the single vehicle simulation requires a small quantity of computational resources, thus reducing the cost of a simulation test.

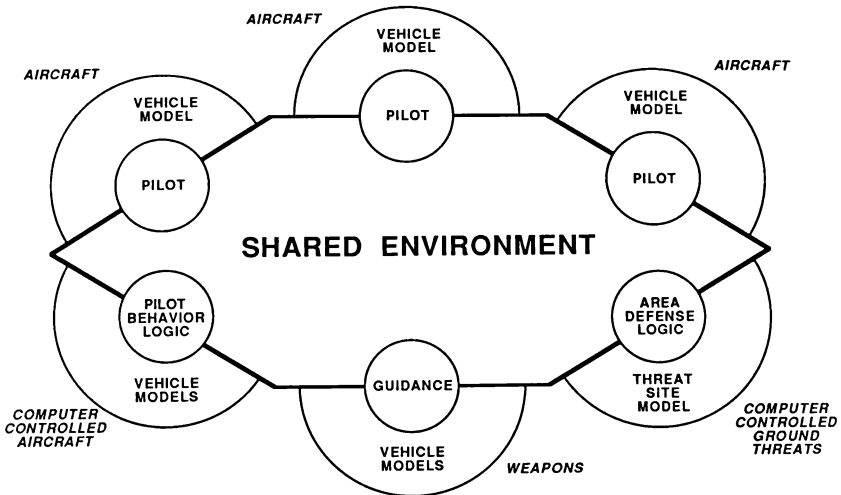


Figure 2. The Multiple Participant Simulation

However, when a simulation study requires a robust environment model containing many complex combat elements, the computational resources of the single vehicle simulation are often inadequate. The typical single vehicle simulation is built around a single computer, or group of closely located computers connected through shared memory. As combat elements are added to the environment model, the computational loading must be constantly adjusted to minimize timing fluctuations. Otherwise, the fluctuations may cause the simulation to become indeterminate and the data collected to be invalid.

Multiple Participant Simulation

The single vehicle simulation cannot support the evaluation of advanced technology in missions requiring cooperative operations. To develop this technology the environment model must facilitate the realistic interaction of multiple manned vehicles. Conceptually, this requires the "opening up" of the single vehicle environment model into a shared environment. Figure 2. (The Multiple Participant Simulation) depicts the concept of a shared environment. The shared environment allows ground threats, computer controlled aircraft, weapons, and other combat elements to become

independent participants that interact through the shared environment.

To support the development of the multiple participant simulation, the following design requirements were established for the shared environment architecture:

- o It shall be compatible with the ITDL physical constraints - multiple securable laboratories, connected via fiber-optics communications.
- o It shall meet real-time criteria (including adequate performance for flight control development, aerial refueling, Close In Combat engagements, ...).
- o It shall support simulation studies ranging from part-task/sub-system development to full-mission/complete-vehicle simulations.
- o It shall provide a progression path from single vehicle simulations to multiple participant air combat simulations.
- o It shall support simulations containing flight ready hardware (hardware-in-the-loop).

The multiple participant simulation supports the interaction of multiple manned vehicles in modern air combat. It focuses on the interaction of the combat elements, and is used to support the development of

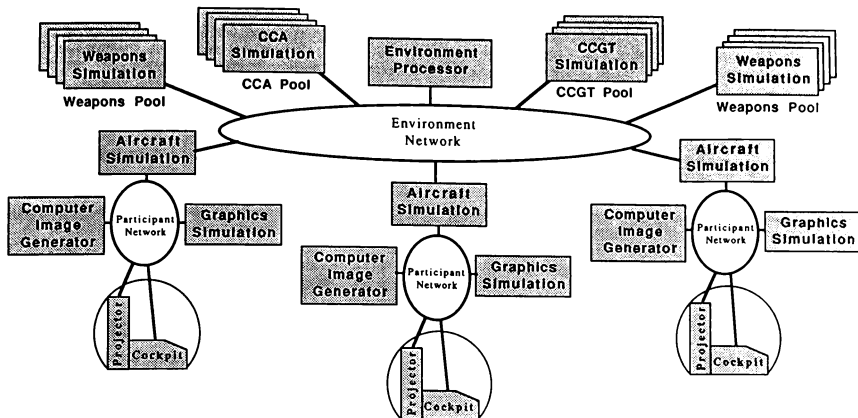


Figure 3. The HIFEN1 Network Configuration

new aircraft technology that aids the crew in managing the complexities of modern air combat.

Proof of Concept

To determine the feasibility of a multiple participant simulation, prior to moving into the ITDL facility, a "proof of concept" demonstration was conducted. The objective of the demonstration was to show that two manned aircraft simulations could interact, in real-time, with one another. The demonstration used two existing single vehicle simulators in a Beyond Visual Range (BVR) scenario. One aircraft simulated a 1980s era fighter with rudimentary avionics. The opposing aircraft simulated a year 2000 concept fighter with advanced avionics. BMA's first multiple manned participant simulation was successfully demonstrated. The next step was to design and implement a fully functional version of the multiple participant simulation in the new ITDL facility.

High-Fidelity Environment One (HIFEN1)

The ITDL's first generation implementation of a real-time, multiple participant air combat environment simulation is referred to as HIFEN1. The HIFEN1 requirements evolved from the design requirements

established by the shared environment architecture. HIFEN1 was designed to provide a multiple participant simulation in which many manned and unmanned participants could interact. HIFEN1 was designed to support Beyond Visual Range (BVR), Within Visual Range (WVR), and Close In Combat (CIC) scenarios. The multiple participant simulation defined by HIFEN1 was to become the foundation for engineering tests, engineering research and development, and concept evaluations conducted at the ITDL.

The HIFEN1 architecture is composed of a centralized Environment Processor, participants, and two types of networks. As shown in Figure 3. (The HIFEN1 Network Configuration), the Environment Processor and all participants (aircraft, ground threats, and weapons) communicate through the Environment Network. Participant sub-systems communicate through Participant Networks.

HIFEN1 Networks

The HIFEN1 architecture uses networks to provide a logical and physical partitioning of simulation data. The logical partitioning of simulation data allows the separation of internal and external participant information. Internal participant information is data which is relevant only to a single participant. External

participant information is data necessary to describe one participant to other participants interacting in the shared environment. The physical partitioning of the participant data allows network loading to be controllable and deterministic. By partitioning the participant's data, system debugging and troubleshooting is simplified.

The Environment Network, as seen in Figure 3, is the medium for the exchange of external data among participants in the shared environment. Data describing participant location, orientation, emissions (e.g., radar, infrared, radio), and other environmental attributes are provided by each participant to the shared environment. In turn, the centralized Environment Processor provides additional data to all participants which describes the relative geometry and sensor signatures of all participants relative to all other participants. Participants extract other participants information from the Environment Network.

The Participant Network, as seen in Figure 3, is used to exchange internal participant data among sub-systems within a participant. Depending on the complexity of the participant, the Participant Network may be physically implemented as a shared memory partition, a multi-processor bus, or a network. Complex participants, such as a manned aircraft, require a Participant Network composed of one or more physical networks. The quantity and composition of the internal participant data varies depending on the subsystems used by a participant.

The distributed architecture of HIFEN1 was designed to use the fiber-optic communication capabilities of the ITDL facility, allowing the ITDL to define a broad range of simulation configurations. The configuration shown in Figure 3, is one of many potential configurations. In addition to physically reconfiguring the Environment Network, participants may be added and removed easily from the Environment Network as scenarios dictate. Also, participants can be run as single vehicle simulation, thus retaining the strengths of the single vehicle simulation. The HIFEN1 architecture provides for flexible, reconfigurable use of ITDL flight simulation resources.

HIFEN1 Environment Processor

The HIFEN1 Environment Processor provides a centralized location for the operator interface and the computation of commonly used environmental parameters.

The HIFEN1 operator interface provides the operator with the ability to control a multiple participant air combat simulation from a single terminal. Scenarios can be initialized, data monitored and modified, and the simulation state controlled. Scenario files specify

participant quantities, model types (e.g., F-15, MIG-23, AIM-120, AIM-9, SA-6), starting locations, and orientations. In addition to the single terminal, a graphical plan view display is available. This display can be zoomed and panned and shows the type, side, location, and orientation of all participants.

The Environment Processor provides data to all participants - data that describes the relative geometry and sensor signatures of participants with respect to all other participants. The Environment Processor gathers the external participant data from all participants in the shared environment. From this data the Environment Processor computes relative geometry data, in the inertial coordinate system and the body coordinate system, for all participants. Using relative geometry, the Environment Processor uses look-up tables to determine the sensor signatures (e.g., radar cross section, infrared) for all participants.

HIFEN1 Participants

The HIFEN1 architecture defines a participant as any combat element interacting in the shared environment. The HIFEN1 architecture assures that any combat element in the shared environment can fully interact with all other combat elements. In HIFEN1, three categories of participants were defined: aircraft, ground threats, and weapons. The aircraft and ground threat categories are composed of both manned and unmanned participants. However, manned ground threats are not currently use in the shared environment. The HIFEN1 architecture allows multiple participants to be grouped on a single processor and communicate with the shared environment through a single interface. In the following paragraphs, manned aircraft, unmanned aircraft, ground threats and weapons will be discussed.

A typical manned aircraft is a high-fidelity six-degree-of-freedom simulation "flying" in a dome. As seen in Figure 3, an aircraft in a dome requires a Participant Network to communicate with the various simulation and aircraft sub-systems. Simulation sub-systems may include in-cockpit graphics generators, out-the-window image generators, target projectors, and a cockpit simulator. Aircraft sub-systems may include avionics, flight controls, and other sub-system modules. When flight ready hardware is required in the aircraft simulation it is connected to the host computer by a Participant Network.

Unmanned aircraft are referred to as Computer Controlled Aircraft (CCA). These aircraft are controlled by pilot behavior logic using planned flight paths and mission goal criteria. To maximize the use of the available computational resources, CCAs are grouped and assigned to a single processor or group of processors. This permits the pooling (grouping) of CCAs. A CCA pool manager is assigned to provide

the interface with the Environment Network. The pool manager controls the execution of all CCAs in a pool.

Computer Controlled Ground Threats (CCGT) provide Surface to Air Missile Sites (SAMS) and Anti-Aircraft Artillery (AAA) sites for area defense of friendly and hostile territory. Like CCAs, CCGTs can be grouped and assigned to a single processor or group of processors. The CCA and CCGT pool managers service requests for activation and deactivation of participants during scenario initialization.

HIFEN1 defines weapons as participants in order to avoid the computational fluctuations that typically occur by having the weapon simulations on the same processor as the aircraft or ground threat simulations. Weapons are activated upon request from other participants. Weapons are grouped and assigned to a single processor or group of processors. However, unlike CCAs and CCGTs, weapons are activated and terminated dynamically throughout a simulation. The weapon pool manager services requests for weapon activation. The weapon pool manager accepts weapon requests, activates the appropriate weapon, and establishes participant to weapon communication (if applicable).

The HIFEN1 environment simulation was completed in 1988. Since that time, it has been used for several research and development studies. The successful use of HIFEN1 in support of Air Force and Boeing projects has provided a full evaluation of the HIFEN1 architecture.

High-Fidelity Environment Two (HIFEN2)

The implementation and operation of the ITDL's first generation real-time air combat environment simulation (HIFEN1) has been a learning experience. The experience gained from HIFEN1 has contributed to the design of the ITDL's second generation real-time air combat environment simulation, known as HIFEN2. The design objectives of HIFEN2 were to take advantage of the following lessons learned from HIFEN1:

- o Computational resources - minimize the computational resources required when expanding a single vehicle simulation to a multiple participant simulation, thus allowing better resource usage.
- o Universal data format - define a universal data format for the Environment Network to allow a broader range of computers.
- o Operator interface - build a more robust operator interface to provide greater simulation awareness.

- o Execution rate - allow any participant execution rate, thus supporting the use of flight ready hardware.
- o Track file data - enhance the Environment Network definition to include track file data, thus allowing sensor errors to be simulated.
- o Data collection - provide centralized data collection to simplify the data collection and reduction tasks.

In the following paragraphs, each of the HIFEN1 lessons learned will be explained, and the HIFEN2 solution summarized.

Computational resources - a significant increase in computational resources, in the HIFEN1 architecture, is required to expand a single vehicle simulation to a multiple participant simulation. Additional computational resources must be allocated for the centralized Environment Processor, as well as for each additional participant or participant pool. The HIFEN2 architecture has distributed the Environment Processor functions among all participants, thus reducing the computational resources required to expand a single vehicle simulation to a multiple participant simulation.

Universal data format - the HIFEN1 architecture does not define a universal data format for the Environment Network. HIFEN1 was implemented on compatible computers, therefore it was acceptable to use the computer's native floating point format. To extend the capability of HIFEN1, HIFEN2 did not assume compatible computers. A more universal data format definition was established for the HIFEN2 Environment Network. The network interface definition requires all data to be represented in a scaled integer format, thus simplifying the network interface for dissimilar computers.

Operator interface - the simple graphical plan view display in the HIFEN1 architecture provides limited simulation awareness. Simulation awareness is necessary when running a multiple participant air combat simulation. HIFEN2 provides an operator interface hosted on a graphics workstation. The workstation allows the operator to have multiple windows into the air combat simulation. Each window can display a two-dimensional plan view, a three-dimensional pictorial view, parametric data, or simulation and participant status. The windows displaying two-dimensional plan views and three-dimensional pictorial views can be panned and zoomed to achieve any desired viewpoint. All windows can be replicated and sized.

Execution rate - the HIFEN1 Environment Network definition restricts the number of valid participant execution rates. All participants are required to

synchronize to the Environment Processor's synchronization pulse. Participants can execute at the Environment Processor's pulse rate or at a sub-multiple of the pulse rate. For example, with a pulse rate of 50 milliseconds(ms), valid participant rates are 5, 10, 25, and 50 ms. As part of the shared environment architecture it was established that the shared environment shall accommodate participants using flight ready hardware. The use of flight hardware, in a participant, requires greater flexibility of participant execution rates. The HIFEN2 interface definition supports a broader range of participant execution rate.

Track file data - the HIFEN1 architecture provides each participant on the Environment Network the "true" location of all other participants in the simulation. There is no provision made for track file data to be exchanged between participants. HIFEN2 has provided a definition for exchanging track file data, thus allowing target tracking data (with sensor errors) to be exchanged between participants.

Data collection - the product of most research and development simulations at the ITDL facility is data. In the HIFEN1 architecture, external participant information was collected by the HIFEN1 Environment Processor. However, internal participant information had to be collected on the participant's host computer. This made data collection and correlation a time consuming task. The HIFEN2 architecture allows a data collection computer to be placed on the HIFEN2 Environment Network. This computer can selectively extract external participant information, as well as internal participant information sent by a participant's host computer.

In addition to benefitting from the HIFEN1 lessons learned, HIFEN2 focused on improving the participant quantities, network and sensor fidelity, countermeasures, and interface gateways to other environment simulations.

Conclusion

The evolution of BMA's real-time air combat environment simulation has provide an opportunity to use the capabilities of the ITDL. The ITDL's independent laboratory design allows the shared environment architecture to retain the strengths of the single vehicle simulation, as well as support the flexibility of the multiple participant simulations.

A long-term strategy has been put in place to capitalize on the lessons learned from previous generations of HIFEN and to extend and enhance the capabilities of each new generation. The development of HIFEN1 grew out of the experiences and lessons learned from the single vehicle simulation. HIFEN2 has benefitted from

the lessons learned in HIFEN1. Today, these real-time multiple participant air combat simulations (HIFEN1 and HIFEN2) provide flexible, reusable tools for the integration and evaluation of advanced aircraft technologies in modern air combat missions requiring cooperative operations.

Acknowledgments

Special thanks to Mike Warden for his many evolutionary observations, and his willingness to support our learning efforts.

References

- [1] David Harvey; The "Virtual World" of SIMNET; Aerospace & Defense Science, May 1990, pages 46-48
- [2] Gary R. George, Samuel N. Knight and Edward A. Stark; Fidelity Requirements in Aviator Training Networks; AIAA Flight Simulation Technologies Conference, Boston, Massachusetts, August 14-16, 1989
- [3] Gary W. McDonald, Robert F. Broeder and Richard J. Cutak; Multi-Ship Air Combat Simulation; 11th Interservice/Industry Training Systems Conference Proceedings, Fort Worth, Texas, November 13-16, 1989
- [4] Summary Report, The Second Conference on Standards for the Interoperability of Defense Simulations; Orlando, Florida, January 15-17, 1990
- [5] Dorsey B. Smith and John J. Soderberg; Real-Time Tactical Simulation for Weapon System Development; AIAA Flight Simulation Technologies Conference, August 14-16, 1989/Boston, Massachusetts
- [6] Jamison Luhn; Tactical Air Combat in a Real-Time Multiple-Engagement Simulation; AIAA Flight Simulation Technologies Conference, September 7-9, 1988/Atlanta, Georgia
- [7] Eric J. Lerner; Avionics that beat the numbers; Aerospace America; May 1988, pages 48-52
- [8] David M. Draffin and John S. Bacha; Implementation of a Secure Multi-Project Laboratory Facility; AIAA Flight Simulation Technologies Conference, September 17-19, 1990/Dayton, Ohio

AN INTERACTIVE COMPUTERIZED AIRCRAFT MODEL FOR REAL-TIME TACTICAL SIMULATIONS

William A. Clark*
Northrop Corporation, Aircraft Division
Hawthorne, California

ABSTRACT

At Northrop Corporation's Aircraft Division in Hawthorne, California, the Flight Simulation Laboratory (FSL) has developed a high speed Interactive Computerized Aircraft Model (ICAM) for use in real-time multiple engagement combat scenarios. This ICAM model allows for a wide variety of mission profiles to be represented. Specific mission types are tailored through user-defined mental data files which define which mental courses of action may be taken and establishes a variety of tactical parameters. This model is virtually indistinguishable from manned participants in that all data reporting and simulation display models treat ICAM's as full manned participants.

ICAM's utilize several lab standard models and data files to help ensure consistency with manned players, and can easily replace, or be replaced, by manned stations or domes. The model interfaces with a wide variety of simulation models in precisely the same manner as manned players. The modular and generic nature of the model source allows for development and testing in an unclassified environment.

INTRODUCTION

In early 1989, Northrop Corporation's Aircraft Division Flight Simulation Laboratory, increased the full participant level from 9 to 20. This included the capability of simultaneously linking 11 Manned Interactive Control Stations (MICS) and 2 domes in an Advanced Manned Interactive Tactical Air Combat Simulation.¹ With the increased participant level, several player slots were made available for digital models. A variety of mission types were identified as needed; Launchable Decoys, AWACS, Jammers, Attack Bomber/Strikers, Ground Control Intercept (GCI) Fighters, and Escort Fighters. A partial list of design requirements include:

- o Act in a tactically realistic manner;
- o Minimum amount of real-time computations;
- o Data file driven;
- o Single model to support all mission types;
- o Use existing lab models where applicable;
- o Modular, flexible, easy to modify;
- o Appear to the rest of the simulation exactly like a fully manned player;
- o Able to replace and be replaced by manned participants;
- o Able to be brought "back-to-life" during a trial as a different type of player;
- o Able to launch and support all missile types;
- o Able to act cooperatively with other ICAMS;
- o Contain multiple, data-driven simplified airframes;
- o Initialize the same as manned participants;
- o Able to act autonomously with or without manned intervention.

A model was then developed using modular structured FORTRAN to include all mission types (Figure 1). The following program elements illustrate the basic model flow:

- o Executive - Calls major routines at various frame rates;
- o Sensor Routines - Bring in team and EOB communication while also establishing ownship detections and tracks;
- o Mental Model Routines - Decide a primary course of action based on sensor routine inputs;
- o Steering Routines - Command inertial guidance commands to flight routine based on basic mental course of action and current tactical situation;
- o Weapon Routines - Select, fire, and support best weapon;
- o Flight Routines - Using steering inputs, airframe state variables are updated and current state of g-LOC (g induced Loss of Consciousness).

*Senior Engineer

Copyright 1990 by Northrop Corp.
Published by the American Institute
of Aeronautics and Astronautics, Inc.,
with permission.

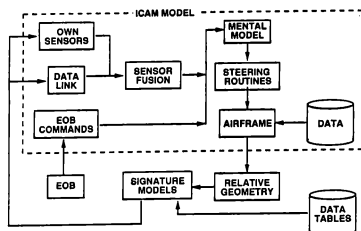


Figure 1

The ICAMS were developed to fulfill a broad range of mission options. The tactics employed are based primarily on observations of manned piloted behavior. Since ICAMS may be assigned to Red, Blue, or both teams, tactics must be consistent with all Rules-of-Engagement (ROE's). Maintaining high speed and consistency with manned models have remained primary design objectives. The greatest challenge for the ICAM model is to make tactically intelligent mental decisions.

EXECUTIVE MODEL

The Executive Model is the key to running the ICAM model fast. All major model elements are run at varying user-defined frame rates. By holding medium to high period functions at a constant frame rate, the Executive is able to stagger calls such that no two high periods are called during the same frame. The desire is to distribute calculations uniformly over a large number of frames. At the same time, we don't want to execute functions any more than we have to. The ICAM approach is to call all functions with a period of 0.5 seconds (or lower) at the user defined frame rate, no staggering allowed. Longer period functions are staggered to evenly distribute computations. The initiation of the calling sequence is randomly started in the event multiple ICAMS reside on a given CPU. This decreases the likelihood of high period functions being concurrently calculated.

The Executive also checks to see if the Test Director wishes to bring the ICAM "back-to-life." The ICAM can be moved instantaneously to a new position in space, to the top of a route, or to a position relative to another player. The Executive may also bring the ICAM "back-to-life" as a different type of player (e.g., bomber, escort, etc.). The result is a force multiplier effect that allows the simulation to appear to have more than 20 participants. Bringing players back to life as different types allows the introduction of "fog-of-war" players later in the trials as other digital players die.

SENSOR MODELS

Sensor Models include own-team contacts, Electronic Order of Battle² (EOB) commands, element leader commands, and own-ship contacts. ICAMS assume perfect IFF knowledge of all contacts.

OWN-TEAM CONTACTS

ICAMS can receive contact data from own-team members (manned or unmanned) in three user-defined modes. First, is a full data link capability with all team members, regardless of range. Secondly, data may be obtained from own team members within a user-defined data link range; and finally, data may be obtained from a paired element member only. The types of contacts reported through data links include visual contacts, radar contacts, radar strobing of team members, lock-on of team members, and inbound missile launches on own-team. Depending on the type of contact, target data is saved for use in the master-track-file.

EOB COMMANDS

EOB may choose to allocate an ICAM to pursue a threatening zone track. If it does, an intercept steering dot is sent along with a last known target location and intercept speed. The ICAMS employ "steal-the-dot" logic for paired interceptors. If the wingman is sent an intercept, and the leader does not have one, he "steals" the dot from his wingman. If the leader and wingman are assigned different intercepts, the wingman tosses out his intercept and uses his leader's.

ELEMENT LEADER COMMANDS

If two ICAMS are assigned to fly cooperatively as a two ship formation, the element lead may send commands to his wingman. Current commands include, Go Autonomous, Fly Formation, and Execute Tactic.

OWN-SHIP RADAR DETECTIONS

Own-ship radar contacts are established through lab standard radar detect routines. A variety of radar capabilities are available and may be changed trial to trial. The ICAMS use their radars in an ESA-like manner. Instead of manually adjusting a mechanically gimbaled Field-of-Regard (FOR), the ICAM files all detectable targets who fall within the maximum radar Field-of-View (FOV). Analysis has shown that this affords the ICAMS a 40% better chance of detecting a target than a manned player who must manually adjust his radar with the maximum FOV. Provisions exist to degrade detections to account for this disparity.

OWN-SHIP E/O DETECTIONS

Electro-Optical (E/O) or Infrared (IR) detections are established through lab standard E/O detection routines. Like radar, one of several different E/O capabilities may be selected from trial to trial. Also, like radar, the I/O sensor is treated as a "staring" detector that files all E/O detections within the maximum FOV. This approximation has far less tactical impact in comparing with manned players than the radar. E/O detections reveal only relative azimuth and elevations to potential threats and are therefore not used as inputs to master track files.

VISUAL DETECTION

ICAMS currently employ a simple method of visually acquiring targets. If a target exists within user defined cockpit masking angles and falls within a user-defined range, the visual detection is made. The lab standard visual acquisition model is a unique and complex model whose use for ICAMS was discarded due to real-time considerations.

MASTER TRACK FILES

The contacts made through the sensor routines are used to build up track files representing the best known information on a target. Sensor data is "fused" using a priority system where more accurate sensors have priority in establishing the track file. For example, if a target was detected both by own-ship radar and from locking up a team member, the master track file would represent own-ship's radar data, since lock-ons give only position.

MENTAL MODEL OVERVIEW

The master track files and other sensor inputs are used by the mental model to determine the general course of action. The ICAM mental model is a data file driven heirarchical, rule-based algorithm. In essence, the data file for the mission type defines which mental courses of action the ICAM can consider. These mental courses of action are always a subset of the total available. The ICAM periodically evaluates all tactical options which he is allowed to consider and saves all actions which apply to the current situation. Using a heirarchical, prioritized sort, the highest priority action is chosen to be executed.

The prioritized list is:

- (1) Dodge active missile
- (2) Dodge Surface-to-Air missiles
- (3) Avoid Forward Edge of the Battle Area
- (4) Avoid high threat targets
- (5) Return to base
- (6) Support wingman
- (7) Follow EOB intercept
- (8) Attack targets
- (9) Attack E/O detect
- (10) Escort team member
- (11) Follow way points
- (12) Orbit Combat Air Patrol point

The mental model also updates master tracks when contacts are lost. If sensor contact is lost, the mental model linearly extrapolates at target position in an attempt to anticipate future target location. After a user defined period of time with no new contact, the target is then "forgotten" and the master track file is dropped.

COURSES OF ACTION AND STEERING

DODGE ACTIVE MISSILE

If an active missile is detected against own-ship, an attempt is made to evade. The ICAM initiates a steep dive at maximum speed while steering to put the missile on the beam. When the missile advances to within about 1.5 nm, a break into the missile is executed in an attempt to out maneuver the final end-game. If the missile is successfully defeated, the ICAM turns and runs from the last known missile heading for 15 seconds, then executes a 15 degree check turn towards the suspected launcher while running for another 15 seconds.

DODGE SAMS

If a SAM launch is detected against own-ship, a dive at maximum speed is made. The ICAM steers directly away from the launcher site in an attempt to out run the SAM. Current SAMS modeled are semi-active beam riders, so actual SAM missile position is not known and therefore an end-game break is not possible.

FOLLOW EOB INTERCEPT

EOB may elect to allocate an ICAM to intercept a zone level track. The ICAM will follow the EOB command flight path at the commanded mach. When approaching within a user defined range of the suspected target location, the radar is turned on and the ICAM converges on the target point. If a target is found near the suspected location, normal attack steering is initiated.

ATTACK TARGETS

ICAM's regularly evaluate all master track files to find who is the highest threat relative to own-ship. Threat value is a function of closure and range. If the highest threat is scored above a user defined threshold, the target is selected to be attacked. Attack steering is engaged in three phases. If the target is beyond 40 nm, the ICAM steers directly at the target. For generally head-on engagements between 15 and 40 nm, a random relative offset of 0 to 45 degrees is initiated to confuse intentions and generate offset for weapons launch. Inside of 15 nm, a lab standard missile steering dot algorithm is used to generate steering commands for missile launch. If during the attack sequence the ICAM is required to make a large heading change, a high or low yo-yo will be employed to turn at the best cornering speed. If the ICAM finds himself attacking from a tail chase position, he closes to just outside minimum launch range.

ATTACK E/O DETECT

ICAM's may wish to pursue an E/O Line of Site (LOS) if no other track information is available. Normally, pursuit will begin with the radar off. The ICAM is looking for a high relative LOS rate to indicate that the target is close. Every 10 seconds, steering commands are updated to put the target 20 degrees off the nose. When the relative LOS rate exceeds 1.5 degrees/sec., the ICAM goes radar active to find the target.

ESCORT TEAM MEMBER

Escorting is the default course of action for an escort fighter. The escort merely flies a loose formation on a player to be escorted. Range and angle off the nose of the escorted member are user-defined. If the escorted player is killed, the escort searches for the closest bomber type and switches to escorting him.

FOLLOW WAYPOINTS

Following waypoints is the default maneuver for bomber/striker mission types. Fifty routes of up to 20 waypoints may be defined, where each waypoint is a 3-D location in space. Commanded mach between sequential way points may be specified. Anti-ship missile launches and/or radar jammer modes may be initiated at each way point. A unique feature of waypoint/steering is the ability to "cut the corner." If, while following a route, an ICAM may deviate from his path (due to missile launch, avoiding targets, etc.). When he decides to return to following his route, a check is made against angle θ (Figure 4).

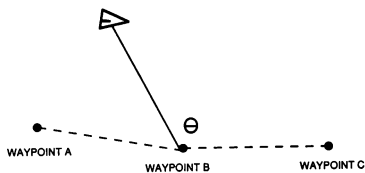


Figure 4

If θ is below a user specified angle, waypoint B will be skipped, and the ICAM will press on to waypoint C. This logic allows ICAM's to skip waypoints no longer tactically relevant.

ORBIT CAP POINTS

Orbiting a Combat Air Patrol (CAP) point is the default maneuver for GCI fighters. The point to orbit is chosen by the user and may be manually re-assigned by the test director during the trial. The ICAM's orbit in a user-defined race track along the expected threat axis. To maximize sensor coverage in the direction of anticipated threats, ICAM's fly slowly towards threats and quickly away from them.

AVOID FEBA

This ability was developed primarily to keep red GCI fighters away from blue SAMS when an ICAM approaches to within a user-defined range of the FEBA. If a target is being engaged while the ICAM approaches the FEBA, the ICAM will turn to parallel the FEBA in such a way as to keep the target abreast on the other side of the line. This allows the ICAM to loiter just inside his region while waiting for an opportunity to re-engage. If the ICAM is chased over the FEBA by missile launches, he immediately runs back over onto his side of the line once the missile is successfully avoided.

AVOID HIGH THREAT TARGETS

Fighter mission types avoid targets quite differently than bomber types. Fighter target avoidance is designed to extend the ICAM away from a clearly defensive position while bomber types merely "stiff arm" or turn slightly away from potential threats in an effort to reduce closure. If a fighter has a target in a 45 degree cone of his tail, at a relatively close range with closure, the fighter will push over to 0 g's and extend away from the target at high speed for 30 seconds. A 15 degree check turn into the target is also initiated to set up re-engagement. If a bomber encounters a relatively high threat target in his forward hemisphere, he initiates a 15 degree check turn away from the target, allowing him to still basically maintain his desired waypoint course.

RETURN-TO-BASE

If a fighter type finds himself winchester (out of missiles) or below a user-defined bingo fuel percentage, the ICAM will begin to head towards the nearest own-team airbase. A descent to 1,500 feet AGL is initiated and the ICAM will loiter above the airfield until removed by the Test Director from the trial.

SUPPORT WINGMAN

This family of actions are used by cooperative fighters assigned in two ship elements of wingman and leader. Normally, the wingman is tied to the leader who issues all commands for the element. The leader will set the wingman free to go autonomous if he is forced to return to base, while the wingman will automatically go autonomous if the leader is killed.

FLY FORMATION

The default tactic called by the leader is for the wingman to fly formation on him. The formation is user-defined by range and angle off the leader's nose or tail. Formation rejoin and steering works rather well, though heavy maneuvering by the leader can make the wingman appear to Pilot Induced Oscillate (PIO).

DRAW AND BAG

If a flight leader approaches within a user-defined range of a target or EOB suspected target, the leader may call for a cooperative tactic. One tactic is the classic drag and bag (Figure 2). If the ICAM is in a defensive posture relative to the target, the element member, closest to the target drags away from the threat. The other element member attacks, hoping the threat is lured toward the "dragger."

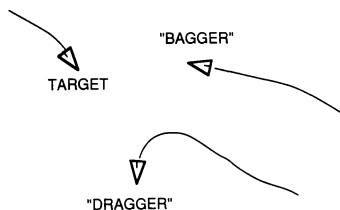


Figure 2

PINCE/SPLIT

Like the Drag and Bag, a Pince/Split may be randomly selected if in a defensive posture. In this tactic, the leader turns to put the target on his beam, while the wingman turns the opposite way to put the target 45 degrees off his nose. When the target is off the leader's beam, both roll back in on the target using normal attack logic (Figure 3).

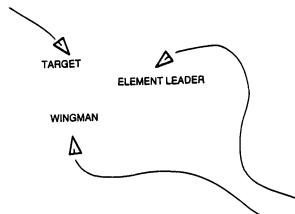


Figure 3

WEAPONS EMPLOYMENT

ICAM's can currently launch and support all available lab missiles including AIM-120, AIM-9L/M, AIM-7F, AIM-54C, and other advanced missiles. Weapon selection is based primarily on target range and weapon availability. Longer range missiles are selected first, until depleted, while shorter range IR missiles are reserved for close-in combat. Within a user-defined range, IR missiles will always be selected over radar missiles unless there is negative closure. In this case, the ICAM drops back to radar missiles in an attempt to make the shot. Lab standard routines of varying fidelity are used to generate minimum, maximum, and maneuver launch ranges (Rmin, Rmax, and Rman, respectively). The ICAM launches at a user-defined percentage of the range distance between Rmax and Rman. For example, a defined percentage of 30% would lead to a valid launch region of Rmin through Rman +0.3 (Rmax-Rman). ICAM's uncage and slew the AIM-9 seeker head to get the necessary self-track tone needed for IR missile launch. ICAM's may also launch autonomous AIM-120 AMRAAMS on visual targets with no radar track. Shoot-Look-Shoot doctrine is employed, with only one missile inbound per target at one time. If a missile fails to successfully intercept, another will be immediately fired, with a maximum firing rate of one missile every three seconds. ICAM's may have outbound missiles on up to 4 separate targets simultaneously.

AIRFRAME MODEL

The ICAM Airframe is a simple 5 degree of freedom point mass model. The ICAM flies along its velocity vector while maintaining coordinated turns in the inertial horizontal plane. Superimposed on the horizontal plane is an inertial vertical pitch axis, thus the inertial horizontal axis is de-coupled from the vertical pitch axis. The airframe is data file driven with a variety of performance types available. User defined parameters include maximum mach (in afterburner and military power), maximum g's, body rates, acceleration, deceleration and altitude. Fuel flow rate and best cornering speed are also specified. The model modifies fuel flow rate, body rotational rates, acceleration, deceleration, and maximum sustainable g's as a function of altitude. Maximum achievable g's as a function of calibrated airspeed is also considered for slow flight. Altitude level-off and ground avoidance is accomplished in the flight routine.

G-LOC MODEL

A lab standard GLOC routine establishes the current state of consciousness based on time aloft and own-ship g's profile. If an ICAM begins to "grey out," he decreases his commanded g's until he is 50% blacked out to allow as much turning as possible. If he inadvertently blacks completely out, no additional steering commands are accepted, and the ICAM drifts along the trajectory established at the time of blackout. After a blackout timer runs out, normal steering is re-engaged.

STATUS DISPLAY

In addition to the standard God's Eye View and parametric displays, there exists a terminal ICAM status display. Invaluable for debugging and real-time monitoring, the display outputs a variety of current state parameters. Display format is cursor-less and constantly refreshing with 4 ICAMs displayed per page. A zoom function shows additional information through a full page format. Status information includes:

- o Team affiliation and member number
- o Mission type
- o Weapon quantity/selection
- o Seeker head state
- o Mach, altitude, g's
- o Radar and jammer on/off
- o Fuel remaining
- o Current course of action
- o Primary target, how detected, time from last detection

AREAS FOR FURTHER REFINEMENT

ICAM's demonstrate that large numbers of tactically effective real-time digital aircraft models can be generated. Currently, ICAM's run at 4-6 ms each on a 5 mip CPU. Although the ICAM's have been judged to be extremely successful, a few weaknesses should be noted. Data structure is not always elegant since the ICAM's are written in FORTRAN. A more modern language such as C or ADA would certainly be useful. Logic or rule based logic for high order mental decisions has its limitations. Value based algorithms simultaneously scoring a wide variety of tactical options would be most effective. ICAM's appear to be superior to manned players in situational awareness while being constrained by more limited tactical options. Over time, manned pilots discern patterns in ICAM behavior and can "game" the system by anticipating ICAM actions. Methods for randomly modifying gains on mental decisions or randomly choosing among a series of tactically equivalent courses of action should be considered. As would be expected, the stricter the ROE's for a mission type, the better the ICAM's appear to perform. As the ROE's become more free-form, making tactically sound decisions in widely varied conditions becomes more difficult.

SUMMARY

Many aspects of the ICAM model have proven themselves quite effective. Using a single, non-reentrant model emphasizing data files and structured code is a must. Interfacing with the rest of the simulation exactly like a manned player allows for a high level of scenario flexibility. Utilizing standard lab models where applicable ensures compatibility with manned players. Judicious subroutine calling helps maintain high speed, which in turn allows for more efficient use of computing hardware.

LIST OF ABBREVIATIONS

AGL	- Above Ground Level
CAP	- Combat Air Patrol
E/O	- Electro-Optical
EOB	- Electronic Order of Battle
ESA	- Electronically Scanned Array
FEBA	- Forward Edge of Battle Area
FOR	- Field of Regard
FOV	- Field of View
GCI	- Ground Control Intercept
GLOC	- G Induced Loss of Consciousness
ICAM	- Interactive Computerized Aircraft Model
IFF	- Identification Friend or Foe
IR	- Infra-Red
LOS	- Line of Site
MICS	- Manned Interactive Control Station
PIO	- Pilot Induced Oscillations
ROE	- Rules of Engagement
SAM	- Surface to Air Missile

REFERENCES

1. Luhn, J. M., "Tactical Air Combat in a Real-Time Multiple-Engagement Simulation," AIAA paper 88-4601-CP, AIAA Flight Simulation Technologies Conference, Atlanta, Georgia, September 7-9, 1988
2. Klos, K. C., "Electronic Order of Battle in a Real-Time Multiple Engagement Simulation," AIAA paper 89-3314-CP, AIAA Flight Simulation Technologies Conference, Boston, Massachusetts, August 14-16, 1989

AT
USAF FLIGHT DYNAMICS LABORATORY

Daniel G. Goddard* and Capt Sheila B. Banks, USAF
Wright Research and Development Center
WPAFB, Ohio

Jeani M. Hackett and John D. Farley
Century Computing, Inc.
Laurel, Maryland

ABSTRACT

One of the missions of the Flight Dynamics Laboratory is to evaluate the tactical effectiveness of emerging technologies for application to new fighters and upgrades to the current inventory. A real-time, multiple pilot-in-the-loop air combat simulation capability was developed to support this mission. In addition to substantial simulator and computer hardware improvements, a software package was developed to provide a threat environment which includes aircraft, weapon, sensor, and electronic countermeasure models. The development of this threat environment, called Tactical Air Combat Software, is the focus of this paper. The Air-to-Air System Performance Evaluation Model (AASPEM) was used as the foundation of the threat environment. The challenge was to convert this non-realtime batch program into a real-time package capable of handling external pilot stations, aircraft, weapon, and sensor models. Through the optimization of code, development of generic interfaces to external simulation models, use of a distributed simulation architecture, and development of a unique data traffic manager, this challenge was met. The resulting combat simulation environment is extremely flexible allowing easy insertion of models of new systems under investigation. A real-time, ten aircraft capability now exists which supports up to six piloted simulators and four digital (computer controlled) aircraft, and provides a comprehensive, expandable air combat environment to support the evaluation of current and future technology integration efforts.

INTRODUCTION

The Wright Research and Development Center's Flight Dynamics Laboratory is currently engaged in several technology integration programs. The Control Integration and Assessment Branch (WRDC/FIGD) supports the Laboratory with high-fidelity, pilot-in-the-loop simulations to evaluate and iterate air vehicle system designs. With the focus of research transitioning to total aircraft system integration, the simulation facility has developed an air combat simulation capability with which to analyze the tactical effectiveness of new air vehicle concepts. This development, referred to as the Tactical Mission Simulation, has involved extensive simulator hardware and computer hardware and software development. This paper focuses primarily on the software development efforts, specifically the development of Tactical Air Combat Software (TACS).

BACKGROUND

FIGD's Flight Control Development Laboratory has traditionally conducted high-fidelity simulations of single aircraft (e.g., F-15, F-16, X-29, etc.) to evaluate handling qualities or other aspects of flight control. However, simulation research is evolving toward a multiple aircraft, full-mission simulation capability due to two important requirements for future aircraft systems: the pressing need to integrate the aircraft's flight, fire, and avionic systems and to develop cooperative air combat tactics. The evaluation of emerging integrated technologies in combat missions which allow forces to fight the threat cooperatively requires a multiple aircraft, full-mission simulation capability.

To transition from a single aircraft simulation facility to a multiple aircraft, full-mission simulation capability, a flexible software environment foundation was needed. Instead of developing a new software environment, FIGD chose to adapt an existing air combat software package for the core of the software environment. Many air combat software programs, including TAC BRAWLER, MIL-AASPEM, ACES, and AASPEM, were evaluated for suitability. The primary selection features were availability and support of the source code, widespread Air Force and DoD use and acceptance, adaptability of the software to FIGD's unique simulation facility hardware architecture, cost and proprietary nature of the software, and functional capability of the air combat simulation. After careful consideration of the various air combat simulations, FIGD chose AASPEM as the air combat software on which to base the simulation environment. The primary features required of an air combat simulation were met by AASPEM: the source code was readily available and supported, AASPEM is used and accepted throughout the Air Force and DoD, the software was not proprietary and could be modified and transported to FIGD, there was no cost to obtain, use, or modify the AASPEM software, and most importantly, AASPEM met the required functional capability for FIGD in-house simulation efforts.

With AASPEM chosen as the core of the software package, the task at hand was to modify it to suit FIGD's particular needs. Due to the extensive code modifications required, the threat environment was renamed Tactical Air Combat Software to avoid confusion with the original AASPEM code.

AASPEM Background

AASPEM was derived from the Piloted Air Combat Analysis Model (PACAM) which began development in 1968. PACAM evolved through a series of upgrades aimed at improving and enhancing the original model. PACAM was first designed to simulate one-on-one aerial combat in three dimensional space.

*MEMBER AIAA

Further developments allowed dynamic reaction to weapon firings, target size variations, bomber penetration, defensive tactics against fighters, and most importantly, multiple aircraft combat. Improvements in model fidelity occurred throughout the many PACAM enhancements. Due to the distinctly separate developments in PACAM, AASPEN is a family of models, each with a specific use and capability, which use the AASPEN "core" to run.

The capabilities of AASPEN provided the framework for FIGD's multiple pilot-in-the-loop, full-mission simulation development. Batch AASPEN can simulate up to 24 aircraft and 75 missiles in-flight. The supported weapons include a long-range radar missile, long-range and short-range infrared missiles, guns, and a CO₂ laser. AASPEN contains a variety of sensors including a searching radar, single-target-track radar, track-while-scan radar, infrared search and track (IRST), radar warning receiver (RWR), laser warning receiver, identify friend/foe (IFF), and a forward looking infrared system. AASPEN's electronic countermeasure (ECM) techniques include noise jamming, deceptive jamming, and blinding. AASPEN's aero-propulsion modeling is a pseudo five degree-of-freedom point mass simulation consisting of a three degree-of-freedom point mass translation plus coordinated turns (roll, then pitch) but no side forces.

Because AASPEN is a batch simulation, it contains pilot decision logic (PDL) to control the digital aircraft participating in the mission simulation. The PDL controls the aircraft in both Beyond Visual Range (BVR) and Within Visual Range (WVR) phases of air combat. The BVR aircraft flight management is controlled by doctrines, rules, and tactics; the fire control is dictated by algorithms and rules; the ECM is controlled by decision tables. WVR tactics are controlled by enhanced maneuver tables which provide realistic offensive and defensive reactions to threats.

AASPEN is used by numerous users throughout the Air Force and DoD community. AASPEN is also supported by a User's Group which meets annually to discuss topics such as current AASPEN applications, modifications, and improvements. Past modifications submitted from the User's Group include new fire control logic, enhanced fighter maneuverability equations-of-motion, and new mission capabilities. The User's Group also sponsors a week-long training course for AASPEN users. Configuration control and distribution of AASPEN has been turned over to the Survivability/Vulnerability Information Analysis Center (SURVIAC).

FEATURES OF TACS

The primary goal of the Tactical Air Combat Software development was to provide a flexible mission simulation architecture that could be quickly adapted for the hardware and software requirements of individual programs. The TACS environment will be used by a variety of programs for analysis activities ranging from total mission analysis to detailed performance analysis of a single aircraft subsystem. In order to provide the degree of support required throughout the 1990s, three major features were incorporated into TACS. First, a modular software structure with standardized interfaces for each type of simulation

component (e.g., aircraft, weapon, or sensor model) was established. Second, the modular software structure was combined with a distributed software architecture that provides a simulation engineer with the capability to quickly map simulation software components onto specific hardware processors. Finally, a distributed heterogeneous hardware architecture was established that allows for mixing a variety of processors and piloted simulators to meet the needs of many different simulation programs.

A general layout of some of the simulation equipment used in the TACS effort at FIGD is shown in Figure 1. The AASPEN software was originally ported onto a VAX 8650 as a non-real-time simulation and extensively modified there before rehosting

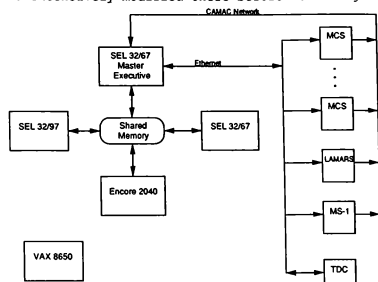


Figure 1. TACS Hardware Architecture Overview

onto the SEL computers. The SELs are interconnected through shared memory to provide a parallel processing simulation environment. Ethernet is used to communicate with non-SEL computers, primarily Silicon Graphics workstations used as simulator display generators for the Manned Combat Stations (MCS), Large Amplitude Multimode Aerospace Research Simulator (LAMARS), Mission Simulator-1 (MS-1), and Test Director Console (TDC). Simulation data input/output (I/O) is accomplished using a fiber optic network, based on the Computer Automated Measurement and Control (CAMAC) IEEE standard, to communicate with the various piloted dome and MCS simulators. Simulation software components are typically distributed around the SEL cluster to optimize efficiency and throughput, as shown in Figure 2. For TACS, a Traffic Manager (TM) was developed to control the flow of data around the computer network.

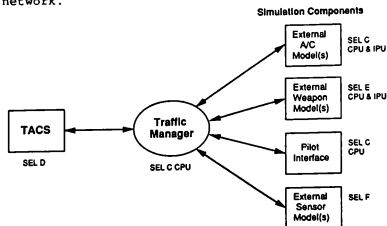


Figure 2. TACS Simulation Component Interface

The following sections describe the major features of TACS in detail, specifically the standardized interfaces, software configuration, and hardware architecture.

Standardized Interfaces for External Simulation Components

The key to providing an extremely flexible mission simulation was to develop a modularized software structure. In the original AASPEM code, all simulation components are linked together in one batch process. Due to the constraints of real-time processing and the need to interchange simulation model components, many of the components had to be separated from the core software and rehosted as processes external to the main core. Therefore the original AASPEM software was restructured to partition the software into logically connected simulation components. This modular structure allows for quickly constructing a custom simulation from "building block" modules. For each component type, a standard interface to TACS was developed and tested. Program specific components under research can be quickly interfaced to TACS via these standardized interfaces. This structure allows the simulation engineer to use standard TACS components for the majority of the simulation system, and use custom-developed models to satisfy specific program requirements. Simulation development efforts can then be focused upon the particular system or mission phase to be studied.

During the TACS development, standardized interfaces were developed for the following simulation components:

- Aircraft Models
- Weapons Models (missiles and guns)
- Sensor Models (RADAR,IRST,IFF, etc.)
- Piloted Simulator I/O

The definition of these standardized interfaces includes all of the information that is required from the specific component to drive other TACS components, as well as all of the information available from TACS that may be required by the component. Simulation components that were available in the original AASPEM software were restructured to communicate with TACS via these standardized interfaces.

External Component I/O. A critical step in the TACS effort was the development of the standardized interfaces for the pilot, aircraft, sensor, missile, and gun input/output components. The goal was to develop well-documented, generic interfaces for each of the components such that existing models could be replaced and new models could be added with a minimum of effort.

An interface design specification for each of these components was produced. The interface design specifications were created by identifying all of the software modules associated with each component. Automated tools were used to identify all of the data needed as inputs to or provided as outputs from the modules. The list of input and output variables was analyzed to identify a subset of the data that was required for proper TACS

operation. This reduced list of input and output data comprised the interface design specification. Complete input/output component interfaces were implemented from these design specifications.

Data Packets. Once the component interfaces were defined, the data packets required to support the interfaces were developed. Data packets, which are simply logical groupings of related data, were implemented as FORTRAN INCLUDE files. One INCLUDE file exists for each packet type. The name of the INCLUDE file, as well as the names of variables and parameters within the INCLUDE file, follow a standard naming convention. Each of the INCLUDE files contain parameters identifying the type of the packet, the length, and the offsets to each field within the packet. Access to the individual fields within a packet are all made via the offset parameters. Consequently, fields can be added and deleted from a packet without having to modify all of the routines which access the packet. To avoid the need for data conversion, all fields within a given data packet are of the same type (e.g., all integer or all real). The primary objective was to organize the information in these data packets to minimize traffic over the Ethernet as well as between TACS and the external component models.

For each simulation component type, a pair of interface routines was developed. The first routine reads all of the TACS-generated data packets required by the external component and places the information into appropriate data structures specific to the component. The second routine obtains output data from the external component's data structures and writes the associated data packets for TACS to use.

Internal to External Component Transition. To support external component execution, a mechanism to bypass the internal TACS model processing was developed. Two new AASPEM-like input forms were developed to define the overall scenario and the aircraft configuration for a TACS simulation. The Aircraft Configuration form contains information which specifies the processor locations of all external and internal TACS component models, as well as the locations of the Manned Combat Stations and the Battle Perspective Display (BPD). With this form, the user can specify the following information for each aircraft:

- Location of the aircraft model component
- Location of the pilot model component
- Number of missiles on the aircraft
- Location of each missile model component
- Number of guns on the aircraft
- Location of each gun model component
- Number of sensors on the aircraft
- Location of each sensor model component

To provide additional flexibility, all component location fields are specified using text mnemonics. The Scenario Configuration form contains information to resolve these text mnemonics into internal TACS data values. Consequently, external components can easily be relocated on another processor by simply changing a single value in the Scenario Configuration form, provided that the component's object code is resident on the desired processor.

Based on the information contained in the Aircraft Configuration and Scenario Configuration forms, data structures were developed to control the execution of the internal TACS components. When an external component is specified, the corresponding interface routine reads the data packet from the component and places the data into the TACS data structure, thereby circumventing the internal TACS component.

Testing and Verification. In order to verify the external component interfaces, a testing strategy was developed. The goal was to verify both the interface itself and the mechanism by which the internal and external components of TACS would communicate. Three separate testing procedures were developed. The first procedure simply tested the validity of the interface to ensure that all of the data required by TACS was being provided. The other two procedures were more complex and verified the communication between TACS and the external component via the Traffic Manager.

A two-step execution method (Figure 3) was used to test the pilot, aircraft, and sensor component interfaces. During the first execution of the simulation, the specific component interface input variables were extracted from TACS for each iteration and placed in a file. During the next execution of the simulation, the data in this file was read back into TACS, thereby simulating the inputs from the external component. The results of the two executions were compared for verification.

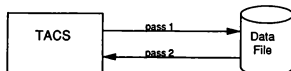


Figure 3. Procedure 1: Two-Step Execution

The second testing procedure simulated the actual software architecture that would ultimately execute on the SEL environment and is illustrated in Figure 4. A VAX terminal-based Manned Combat Station (MCSTerm) was developed to simulate real external pilot inputs (e.g., stick, throttle, missile and gun triggers). MCSTerm provided a display of aircraft and missile positions, velocity, altitude, and maneuver state. In addition, MCSTerm allowed the developer to pause the simulation and single step through iteration cycles. MCSTerm was integrated with an aircraft model on the VAX, providing both an external aircraft component and an external pilot interface for more extensive TACS testing.

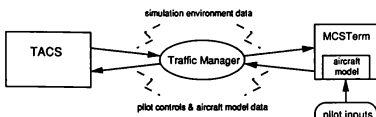


Figure 4. Procedure 2: MCSTerm Verification

The final testing procedure also simulated the distributed software architecture (Figure 5). This procedure involved extracting internal TACS components to act as external components. Extracted components were executed as separate processes on the VAX, and communicated with the main TACS process via the Traffic Manager and a

shared memory partition. This testing method was extremely effective because it allowed for baseline test cases to be verified against a multi-process simulation. This method was used to verify both the missile and gun component interfaces.



Figure 5. Procedure 3: Extracted Component

Distributed Simulation Software Architecture

To fully exploit the modularized TACS structure, a distributed software architecture was established. The architecture allows for easy and flexible distribution of TACS components across the SEL computer cluster. The individual TACS components, both internal and external, are basically "building blocks" which can execute on the various SEL processors. Synchronized communication between the distributed components is accomplished by a centralized Traffic Manager. The TACS modular software structure is augmented by a Test Director Console, which allows the test director to set-up, monitor, and control simulations remotely from a workstation.

Traffic Manager. A key element of the distributed TACS architecture is the centralized Traffic Manager. TM is a generic data packet handler developed to manage communication between TACS, the external components, and the piloted simulators. TM supports communication between the distributed SEL computers via shared and reflective memory, and communication between the SEL and Silicon Graphics computers via Ethernet and CAMAC hardware. Figure 6 illustrates the Traffic Manager and the associated data packets within the TACS system.

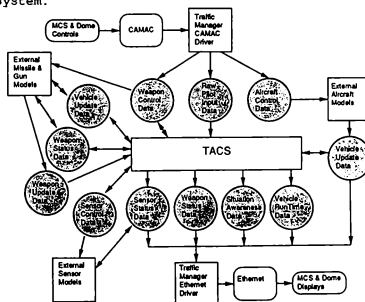


Figure 6. Traffic Manager & Data Flow

To support communication, TM maintains a database of packets containing all of the information for the various TACS interfaces. Each packet is time-stamped to facilitate data synchronization between the TACS processes. TM also provides utilities to read and write packets that are stored in its database. Callers can read and write entire packets or specific portions of packets, called sub-packets. The ReadPacket utility can be invoked to either return the currently available packet data, or wait for an

updated packet based on an input time stamp value. The WritePacket utility allows the caller to specify a priority for the write operation, which controls where the packet is placed in memory.

The TM database is distributed across various common memory buffers, including datapool, extended memory, and reflective memory. Special communication areas are used to facilitate the transfer of data between these common memory buffers. The TM database contains directory entries for each packet indicating the buffer location, address within the buffer, type, vehicle ID, length, update flag, and time stamp for the packet.

A major responsibility of the Traffic Manager is to ensure that data is not corrupted during read or write operations. To meet this requirement, TM limits packet access to a single writer and multiple readers, and uses software locks to prohibit simultaneous reading and writing.

The Traffic Manager supports the piloted simulators by allowing automatic registration for packets to be sent to the MCSs and dome simulators. Once a packet has been registered, subsequent updates are automatically sent via the Ethernet link to the corresponding MCS or dome.

Test Director Console. The Test Director Console provides a user-friendly, flexible interface to TACS. The TDC software enables overall simulation control from a central location, and provides "quick and easy" set-up and modification of TACS scenarios and input parameters. In addition, the TDC is integrated with a Battle Perspective Display, providing an adjustable view of the combat airspace. The TDC also supports a complete data collection and post-run analysis capability. All of these functions can be performed simultaneously using the TDC's window-based interface executive. The TACS TDC consists of three main components: scenario builder, Battle Perspective Display, and data analyzer.

TACS efficiency was severely restricted in its initial AASPEM-like configuration due to the tedious and time-consuming process of developing the large set of scenario specific data input forms. Input form modifications were previously made using a text editor. Misplacing a single character by one column in an input form could result in a severe failure during simulation execution. In order to simplify the task of scenario development and testing, a scenario builder was implemented.

The TDC scenario builder provides a simple graphical interface allowing the test director to specify and control TACS scenarios. This interface allows the test director to choose aircraft, missiles, guns, etc. from standard libraries of forms using menu selections. Aircraft can be selected and then placed on a strategic map to specify their initial locations. Individual aircraft can be loaded with missiles, gun rounds, and laser fuel. The scenario builder also provides the capability to select and modify overall simulation parameters such as tactical doctrines, penetrator waypoints, electronic warfare options, and a variety of simulation options. In addition,

the scenario builder allows the test director to map the individual TACS software components (e.g., external components) onto specific processors in the distributed computer network.

A major component of the scenario builder is a flexible input form builder. The form builder eliminates the tedious AASPEM-like input forms editing previously required by TACS. The input form builder is a menu-driven interface which generates TACS input forms for aircraft types, missile types, etc., and stores them in libraries. The libraries of forms are then used by the scenario builder to generate TACS scenarios.

The Battle Perspective Display provides an adjustable view of the combat airspace. The display consists of a grid representing the ground, which can be traversed via a flexible input mechanism (mouse inputs, menu selections, and/or keyboard interaction). The input mechanism supports both ground and elevation translations, as well as rotations about the x,y,z axes. Aircraft are displayed on the grid with corresponding aircraft ID, altitude, trailing vector, and ground track. The trailing vector displays the last six seconds of the flight path. Missile trajectories are also displayed. Figure 7 illustrates the BPD for a sample TACS run.

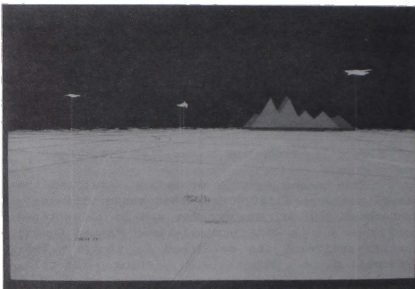


Figure 7. Battle Perspective Display

The BPD provides the test director with a variety of options. The Status window displays information on each aircraft, including aircraft ID, altitude, heading, Mach number, airspeed, and angle-of-attack. The Relative Data window displays information about two selected aircraft such as range, bearing, and angle-off-tail. The Views option provides five different canned viewing angles: top view, southward, northward, eastward, and westward across the grid surface. The Plane View option provides an out-the-window view from any selected aircraft. Other options include zoom, decluttering, and scenario replay.

An important feature of the TDC is the data collection and analysis component. Prior to simulation execution, the test director can specify, from a menu of parameters, the real-time data variables (e.g., aircraft x,y,z positions) and simulation events (e.g., aircraft kills, missile launches, etc.) to collect. The test director can also specify the rate at which simulation data is saved. After the simulation run is completed, the

test director can select an event summary report of the run, scenario information for each aircraft, and data plots of individual simulation parameters. In addition, the data analyzer provides the test director with the following capabilities:

- Compare data variables and events from different simulation runs
- Perform mathematical functions on parameters
- Perform statistical functions on parameters
- Specify user-defined functions to manipulate parameters
- Plot parameters and events from single or multiple simulation runs

Distributed Simulation Hardware Architecture

The task of mapping the distributed software architecture onto the specific hardware of the Flight Control Development Laboratory posed significant challenges. The existing facility contains several million dollars worth of computers and cockpit simulators. The primary concern was to develop a hardware architecture that would exploit as many of the existing resources as possible, while maintaining easy upgrade paths to new equipment and technologies as they are procured. The approach taken was to treat computers and cockpit simulators as "black boxes" with clearly defined interfaces. Industry standard hardware was chosen for all data communication paths between the "boxes". This approach allows for easy insertion of new equipment, or replacement of aging equipment with new technologies.

A hardware block diagram of the current simulation facility is shown in Figure 8. The heart of the facility is the cluster of SEL computers. Each computer contains two processors that typically operate in a synchronous fashion. Specific TACS software components can be directed to a specific processor via the TDC scenario

builder capability. High speed communication between the processors is provided by shared and reflective memory. Communication between the computer network and the piloted simulators is performed by two standard communication networks: Ethernet and CAMAC. These three communication mechanisms (Ethernet, CAMAC, and shared/reflective memory) provide the hardware backbone of the TACS hardware architecture and are discussed further in the following sections.

Ethernet. Due to the availability of Ethernet on virtually all computing equipment, it was chosen as the standard interface mechanism between the SEL computer cluster and all external processors. While Ethernet is not normally associated with real-time operations, data rates analysis of the system determined that Ethernet's 10 Mbps transfer rate could support the anticipated data traffic, provided that packet collisions on the bus could be minimized. In order to reduce bus traffic, special software drivers were developed for each of the computers. These drivers avoid the overhead of layered Ethernet protocols such as TCP/IP. By utilizing the bus at the low-level Ethernet 1.0 specification, all "hand-shaking" and error correction traffic was eliminated. During real-time simulation operation, a lost frame of data is considerably less important than maintaining constant throughput.

Ethernet is not the final solution for TACS inter-processor communications. The Ethernet links will be replaced when higher speed communication devices (e.g., FDDI) become generally available. For now, Ethernet provides a standardized communications mechanism that allows for easy interchange of data between a wide variety of hardware platforms.

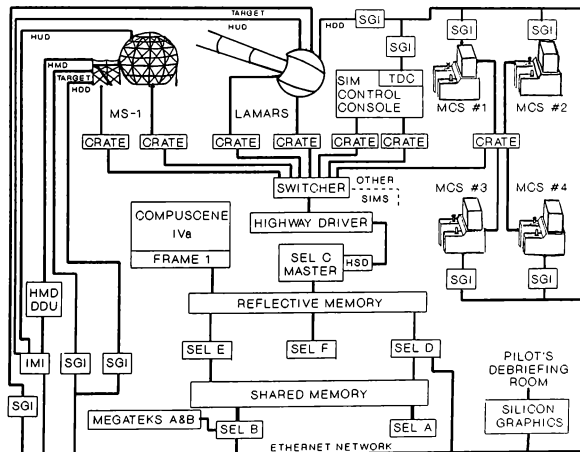


Figure 8. Flight Control Development Laboratory

Shared Memory. While Ethernet provides the mechanism for communicating with the non-SEL processors, inter-processor communications within the SEL computer cluster require much higher data rates. Multi-ported memories are used to connect the six SEL computers in the cluster. High-speed shared memory and reflective memory subsystems are used to meet this requirement. Memory partitions are set up for simultaneous access on all processors. Access to the memory partitions is controlled by the Traffic Manager software to prevent data corruption. The Traffic Manager also provides a timed synchronization mechanism that allows a processor that is receiving data to access shared data when the data is needed, without requiring any handshaking or communications with the sending processor.

CAMAC. A standardized mechanism was also required for interfacing a wide variety of analog (e.g., sticks, throttles) and discrete (e.g., switches) signals with the computer cluster. In order to provide a mechanism that allows for easy substitution of processors or analog components, the high-speed CAMAC network was chosen. The CAMAC network acts like a 24 megabit/second computer channel. CAMAC interface cabinets, or crates, are located at each of the dome simulators, the simulation control console, and the Manned Combat Station area.

A Kinetics System highway driver provides a standard interface between the computer cluster and each crate. This interface allows for substituting another host computer or for adding other computers to the CAMAC network. Data travels from the source computer channel through the serial highway driver, across the network to the addressed crate, through the serial crate controller to the crate backplane, and arrives at the memory of the addressed module. Any computer on the network can receive information from, or send information to, any interface device on the network. Standardized hardware interfaces to major components (e.g., throttles) were also developed so that components in various dome simulators and Manned Combat Stations would be plug compatible for easy modification or exchange.

TACS DEVELOPMENT PROCESS

TACS was implemented as a series of cooperative efforts. The original AASPEM software was converted from a batch, non-real-time simulation executing on a VAX, to a piloted, real-time simulation executing on a distributed SEL computer environment. Manned Combat Stations were developed and integrated with TACS to support external pilot inputs to the simulation. Similarly, the LAMARS and MS-1 dome simulators were also integrated with TACS.

Conversion from Non-Realtime to Real-time

A key factor in the AASPEM-to-TACS conversion effort was the selection of a comprehensive software development environment to support the transition from a non-real-time VAX-based simulation to a real-time SEL-based simulation. One of the main goals of this selection process was to automate the TACS conversion process as much as possible. Other important criteria included strong configuration management, software verification, and data analysis capabilities.

The Simulation/Rapid Prototyping Facility (SRF) was selected as the development environment for TACS. Since the SRF was already hosted on the VAX and AASPEM runs under VMS, the first portion of the AASPEM-to-TACS conversion was handled on the VAX. The SRF environment supports a rapid-prototyping approach to simulation software development by providing a powerful set of capabilities:

- User interface development
- Software development
- Automated software system build
- Configuration management
- Performance analysis
- Simulation time history data analysis
- Code analysis
- Automated test verification
- Automated problem reporting
- Automated software documentation
- Automated VAX to SEL porting tools

VAX to SEL Transport. The first step was to prepare the VAX-based AASPEM software to be compatible with the SEL computer environment. The initial version of AASPEM incorporated a generic data output mechanism for saving simulation data, which allowed for automated test verification against baseline AASPEM test results. This version of AASPEM was tested using a variety of scenarios, 8v7 being the most comprehensive, and was used to establish the baseline execution results for future testing comparisons.

In order to transport the VAX version of AASPEM to the SEL computers, all of the files were converted to the SEL FORTRAN format using the SRF porting utility. Once the files were transported to the SEL computers, several problems were discovered. Minor compilation problems were easily corrected, and the resulting modifications were made to both the VAX and SEL versions of AASPEM/TACS. However, several major SEL compatibility problems were encountered. First, the TACS global data was too large to fit in datapool (shared memory), and therefore had to be placed in SEL extended memory. Consequently, all of the TACS common blocks were mapped into the extended memory address space at specific memory block offsets. In addition, all TACS procedure and function argument lists were modified to allow for extended mode addressing. Automated command procedures were developed on the VAX to perform the necessary modifications and memory mapping. These command procedures were used extensively as newly developed TACS code was transported from the VAX to the SEL.

Second, the TACS code was too large to fit as a single program within the limited SEL task execution space (128K words). Consequently, the TACS software was divided into eight separate tasks. Two SEL versions of TACS were established. A non-real-time version was developed initially and used to perform baseline testing and timing analysis. A real-time version was produced later to support piloted testing and demonstrations.

Task sequencing for the real-time TACS software also presented problems. The SEL MPX operating system services were too slow to be used for a real-time simulation. Consequently, the real-time TACS software used an FIGD-developed "fast overlay"

mechanism for task execution sequencing. This rapid context switching method bypasses the MPX operating system and allows the application task to perform the switch directly.

Verification. Initial verification testing compared the SEL results of both a lvl scenario and an 8v7 scenario to the baseline results. The SEL simulation data was transported to the VAX, and then compared and plotted against the VAX simulation data using the SRF analysis utilities. Minor execution differences existed between the VAX and SEL versions for the lvl scenario. Unfortunately, major differences existed for the 8v7 scenario.

An analysis concluded that the execution discrepancies were caused by floating point precision differences between the two computers. This precision difference resulted in global events being scheduled at different time steps in the two simulation versions, which in turn caused differences in pilot decision logic tactics, maneuver states, x-y goal positions, etc.

To prove this conclusion, minor time increments were added to the major TACS global event scheduling routines on the VAX to counteract this precision difference. The resulting VAX test results for the 8v7 scenario were nearly identical to the SEL test results. Instead of modifying all of the event scheduling routines to counteract the precision differences, a decision was made to establish separate baseline test results for the VAX and SEL versions of TACS.

Optimization. The task of optimizing TACS to execute in real-time was quite a challenge. At the outset of the optimization effort, an estimated 50% or greater speedup was required to provide the minimum real-time capability originally desired, a three piloted aircraft versus four digital aircraft combat scenario. The overall objective of the TACS optimization effort was to maintain compatibility with the baseline test cases. Simulation integrity was a matter of great importance. Consequently, several optimization efforts were discarded because they resulted in execution differences, most notably a rewrite of several trigonometric functions which resulted in an approximate 20% speedup.

TACS optimization was an iterative process. The first several optimization updates concentrated on eliminating stub subroutines and replacing calls to simple functions with references to global arrays. Over 70 routines were eliminated during this process. Other optimization iterations consisted of major rewrites to numerous sensor, missile launch, weapon screening, vehicle state data, and pilot decision logic routines. Significant optimization improvements involved splitting several of the major TACS functions into separate processes executing in parallel on the SEL dual-processor configuration. The split functions included sensor processing, vehicle states integration, and vehicle geometry calculations.

The resulting timing data for the final optimization as compared to the original TACS baseline resulted in a CPU execution time decrease from 2387 seconds to 730 seconds for the 4v4 test case, a 69.4% improvement.

Integration into Simulation Facility

The culmination of the TACS implementation effort was the integration into the Flight Control Development Laboratory. The Laboratory contains a variety of hardware and software elements which reduce the effort of developing and integrating a real-time simulation. The Laboratory provides a common framework for real-time simulation development, including a standardized hardware architecture, communication interfaces, a library of software models, and a variety of hardware components such as terrain visual generators, target projectors, helmet mounted displays, graphics workstations, and piloted simulators. In addition, the Laboratory maintains a suite of software development tools including post-run data analysis, library management, inter-computer file transfer, and remote processor control.

The non-realtime version of TACS was integrated using a fully distributed software architecture that exploits all of the available hardware resources. Existing real-time simulation executive components were utilized as much as possible during the integration.

Simulation Executive. The non-realtime to real-time transformation effort was greatly reduced by using the Standard Simulation Executive. This executive provides a comprehensive, ready-to-use library of real-time simulation capabilities, including:

- Synchronized scheduling of processes on all SEL computers
- Real-time simulation data recording
- Automatic process activation
- Shared memory manipulation
- Real-time simulation data display
- Pause and single step execution modes
- Timing analysis for individual software components

In addition, the Simulation Executive provides templates for building and integrating the individual components of the TACS simulation. These templates were customized for each processor in the distributed SEL cluster.

External Components. TACS was integrated with a variety of external components, including an F-15 aircraft model; U.S. and Soviet missile models; M61A1 gun model; and radar,IRST,IFF, and RWR sensor models. These components were all integrated with TACS using a standard methodology in which the components were synthesized into the Simulation Executive using the appropriate templates. Stand-alone real-time testing was performed to verify each component's execution. Finally, to facilitate proper distribution of the external components, detailed timing analysis was performed. Timing data such as minimum, average, and maximum execution times were obtained for all components in the TACS network. Based on the resulting timing data, the components were distributed to individual processors in the SEL cluster. The Aircraft Configuration forms were updated to specify the use of external components for individual aircraft, as well as the processor locations for each external component. The external components were integrated and tested with the Traffic Manager and TACS using a variety of scenarios.

Manned Combat Stations

Only so much realism can be obtained by a computer controlled aircraft. This limitation is especially evident in a real-time simulation where the amount of code emulating the pilot's decision-making processes is constrained by the frame time. Therefore, involving as many pilots in the scenarios as possible is highly desirable to add credibility to each engagement. To do this in a cost effective manner, FIGD followed the standard industry practice of developing Manned Combat Stations, as shown in Figure 9. A simple wooden "cockpit" was designed and checked for anthropometric acceptance before five cockpits were fabricated. In order to provide the pilot with familiar throttle and stick controllers, authentic F-15E throttles and stick grips were installed. The stick grip was mounted to a joystick controller in a sidestick arrangement. No rudder pedals were included in the MCSs. The interface between the analog stick, throttle, and associated switches and the computers is based on a standard design for all FIGD manned cockpits, either MCS or dome. This commonality eases maintenance and allows swapping components if problems arise. This interface is based on the CAMAC IEEE standard which was discussed in an earlier section.

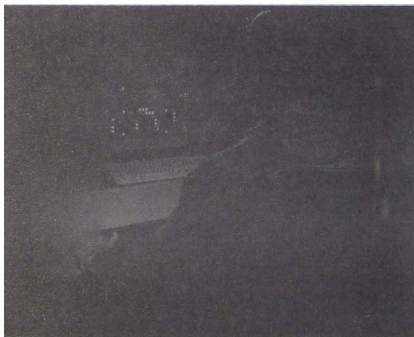


Figure 9. Manned Combat Station

The cockpit displays and out-the-window visual for a MCS are generated by a Silicon Graphics 4D/85GT and presented to the pilot using a 19" color monitor. An advanced, all-glass cockpit with 3 simulated CRTs, a head-up display, and out-the-window visuals was integrated with each MCS. Sensor data is presented to the pilot on a situation display to indicate range and azimuth to targets, as well as target heading. If deemed necessary, additional target information such as altitude, speed, and type of threat can easily be presented to the pilot. The "god's-eye view" display also indicates the position of friendly and neutral forces. These displays are intended to be used during the BVR portion of an engagement. For WVR or Close-In Combat (CIC), an entirely different set of displays was developed to enable the MCS pilots to effectively perform CIC with pilots flying in the dome simulators. These displays use a number of techniques to provide the pilot with 4-

pi steradian line-of-sight knowledge of threat position and attitude. The pilot can switch between several different display formats depending on the given situation and informational needs.

LAMARS and MS-1 Simulators

Both the LAMARS and MS-1 were integrated with TACS to provide high-fidelity pilot stations. The LAMARS, shown in Figure 10, is a beam-type, motion-base dome simulator. Its past applications have concentrated on high-fidelity, handling qualities studies. For its use in air combat simulation, the electronics of the slewable, high-resolution target projector were updated and laser target projectors were added, as well as a helmet mounted display.

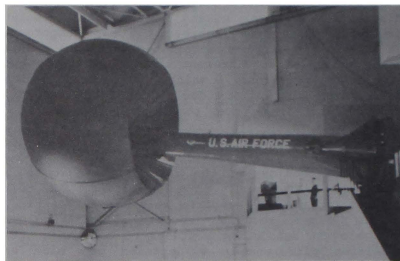


Figure 10. LAMARS Simulator

The MS-1, shown in Figure 11, is a state-of-the-art, 40' dome simulator designed specifically for air combat simulation. The MS-1 features four laser target projector pairs, two background projectors, a high-resolution target projector pair, and a slewable area-of-interest projector. It also can be equipped with a helmet mounted display. The high-resolution target is generated by a Silicon Graphics machine while the background imagery for the MS-1 is produced by a Compuscene IVa image generator. These two dome simulators are typically used as a lead/wingman flight element, although it is possible to conduct lvl engagements between the domes.

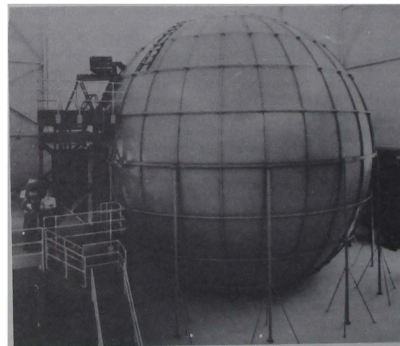


Figure 11. MS-1 Simulator

CONCLUSIONS

Capability Demonstrations

The development of the Tactical Mission Simulation capability, of which TACS is a central part, has progressed in a phased manner. The first demonstration of the air combat capability at FIGD took place in November 1989, which marked the completion of the conversion of batch AASPEM into real-time, piloted TACS. For this demonstration, the LAMARS and two Manned Combat Stations were flown as piloted stations against four computer-controlled aircraft. The demonstration established the soundness of the software architecture design and implementation including the data traffic manager, external aircraft component interface, pilot input/output interface, and code optimization practices. The 3v4 simulation ran in real-time with up to six missiles in flight simultaneously.

In June 1990, a second demonstration was held when the MS-1 was integrated into TACS. In addition to the features of the first demonstration system, the four laser target projector pairs in the MS-1 were driven using TACS threat information, a high-fidelity F-15 model replaced the simpler transfer function model used in the November demonstration, and throughput was substantially increased in the data traffic manager through a software improvement. The use of the MS-1 with its superior visual cues vividly demonstrated the usefulness of the Tactical Mission Simulation.

Scheduled in September 1990, the most recent increment to TACS will be demonstrated. Additions to the simulation capability will include the capability to use external weapons and sensor models, optimized PDL code, and the Test Director Console to set-up, control, and analyze each engagement. This demonstration will exhibit the flexibility offered to the user in setting-up software configurations and combat scenarios. The test conductor will be able to choose from a wide array of aircraft, weapon, and sensor models which can run either internal or external to TACS. Force ratios and piloted stations will be easily changed through the TDC.

Future Directions

During the development of TACS, many potential enhancements to the system became apparent. Current plans to increase the functional capability of the system include the following:

1. Linking to remote simulation facilities was investigated and a conceptual design performed in the development of the original TACS system; however, a remote link was not implemented. The implementation of a link to the WRDC/TXAA MIL-AASPEM system is planned to increase the number of manned players available in the simulation.

2. In the present system, the mix of a piloted lead aircraft and a digital wingman is not supported resulting in an uncoordinated lead/wingman team. A planned enhancement to the system is the ability to have a piloted lead aircraft and a digital wingman aircraft whose maneuvers are coordinated.

3. Test Director Console enhancements are currently planned and include a touch screen, the ability to control the digital aircraft from the TDC, and further Battle Perspective Display and user interface enhancements.

4. Advancements have recently been made in technologies which can be applied to Pilot Decision Logic. Enhancements for TACS PDL, such as the incorporation of a PDL expert system, are currently under investigation.

5. The current TACS simulation system is written in FORTRAN. To take advantage of new aircraft, weapon, and sensor models being developed in Ada, rewriting TACS in Ada is being considered in WRDC.

6. The required computer throughput increases as additional TACS capabilities are added. To maintain real-time operation, further distributed processing must be performed. To maintain flexibility and allow expandability, a variety of hardware platforms will be investigated to implement the software distribution.

7. Upgrades are also planned for the MCSs. Enhanced displays are in development; touch screens and a new communications and sound system are also planned improvements to the MCSs.

Program Applications

The first application of Tactical Mission Simulation and TACS will be for the Integrated Control and Avionics for Air Superiority (ICAAAS) program. In addition to the contractor's ground-based simulation of the ICAAS system, FIGD is providing an independent analysis of the effectiveness of ICAAS's flight and attack management software. The MS-1 and LAMARS simulations will be equipped with the ICAAS capability and evaluated against eight threat aircraft in a variety of scenarios. The eight threat aircraft will be a mix of MCSs and digital aircraft.

Other future technology integration programs such as Multi-System Integrated Control or Integrated Tactical Aircraft Control may make use of the Tactical Mission Simulation capability. In addition, many in-house R&D programs investigating the tactical pay-off of high agility aircraft or high angle-of-attack flight will make use of the simulated threat environment as a cost effective method to gather performance data.

NOMENCLATURE

AASPEM	: Air-to-Air System Performance Evaluation Model
ACES	: Air Combat Engagement System
BPD	: Battle Perspective Display
BVR	: Beyond Visual Range
CAMAC	: Computer Automated Measurement And Control
CIC	: Close-In Combat
DoD	: Department of Defence
ECM	: Electronic Countermeasures
FIGD	: Contr ' Integration and Assessment Branch
I/O	: Input/ Output
IFF	: Identity Friend or Foe
IRST	: Infrared Search and Track

LAMARS : Large Amplitude Multimode Aerospace
 Research Simulator
 MCS : Manned Combat Station
 MCSTerm : Manned Combat Station Terminal
 MS-1 : Mission Simulator 1
 PACAM : Piloted Air Combat Analysis Model
 PDL : Pilot Decision Logic
 RWR : Radar Warning Receiver
 SEL : Systems Engineering Laboratories
 SRF : Simulation/Rapid Prototyping Facility
 SURVIAC : Survivability/Vulnerability Information
 Analysis Center
 TACS : Tactical Air Combat Software
 TDC : Test Director Console
 TM : Traffic Manager
 TKAA : Technology Exploitation Directorate
 WRDC : Wright Research and Development Center
 WVR : Within Visual Range

ROCKWELL'S REAL-TIME SIMULATOR AIDS HYPERSONIC VEHICLE/NASP DESIGNS

by

William G. Burnett*
Rockwell International, Space Systems Division
Downey, California

Abstract

NASA and DOD are considering the development of hypersonic vehicles (HSVs). Candidate HSV concepts cannot be flight tested but must be validated by simulation methods. Issues relating to adaptive guidance, navigation, and control (GN&C) systems and their related man-machine interfaces must be simulated in an integrated, high-fidelity fashion. The present state-of-the-art in HSV simulation is based on the Space Shuttle orbiter and has not advanced significantly since the late 1970s. One of the primary deficiencies is that present methods do not enable real-time HSV simulation and cannot accurately describe the flight qualities of the latest integrated airframe/engine concepts and related display/control systems. A state-of-the-art generic manned hypersonic vehicle simulator is needed to evaluate flying qualities, crew work loads, and flight safety issues. This simulator would also be used to identify future program requirements and perform preliminary assessments of simulation technology capabilities.

This paper provides a synopsis of actual simulation development operations and the methodology for an HSV laboratory, including the system architecture of the HSV simulation itself.

Introduction

The first test flight of the Air Force's new X-30 experimental vehicle will not happen before 1995, but by that time, pilots and engineers will have spent years at its controls and will have flown the X-30 through all of its mission phases. Rockwell has consolidated the major simulation activities for its Aerospace Operations in the HSV Simulation Laboratory (Fig. 1) located in Downey, California, under the direction of A. J. Mauceri. The facility is reconfigurable; it is not built for a specific program, so the company is able to modify its configuration to support both manned and unmanned programs. Rockwell's laboratory has created man-in-the-loop simulations for such programs as Apollo, Skylab, Apollo/Soyuz, Shuttle, Space Station, B-1B, and X-31 and is applying these 23 years of experience to the task of hypersonic flight. A secure area (Fig. 2) within the laboratory has been set aside to support internal research and development.

This secure facility is used to simulate hypersonic vehicles such as the Shuttle, Assured Crew Return Vehicle (ACRV), Advanced Manned Launch System (AMLS), X-30, and National Aerospace Plane (NASP)-derived vehicles. The simulation laboratory is designed to support study areas including the following:

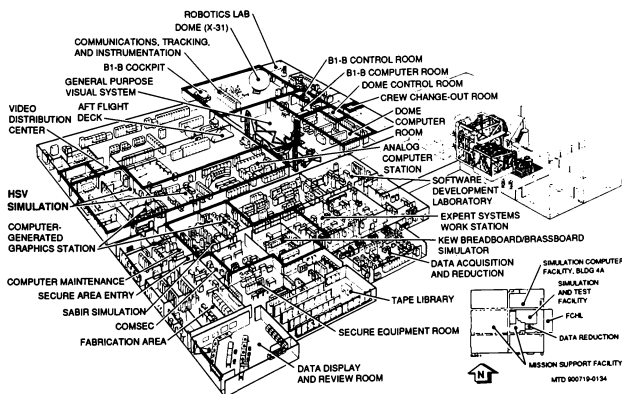


Fig. 1. The simulation and systems test (SST) laboratory is Rockwell's "simulation center of excellence," and supports such major programs as the X-31; SDI/SABIR; B-1B; and hypersonic vehicles development such as the Shuttle, ACRV, and NASP. The laboratory is fully self-supporting, including the design and build of all special hardware/software needed for avionics design, build, test, and integration.

*Member, AIAA

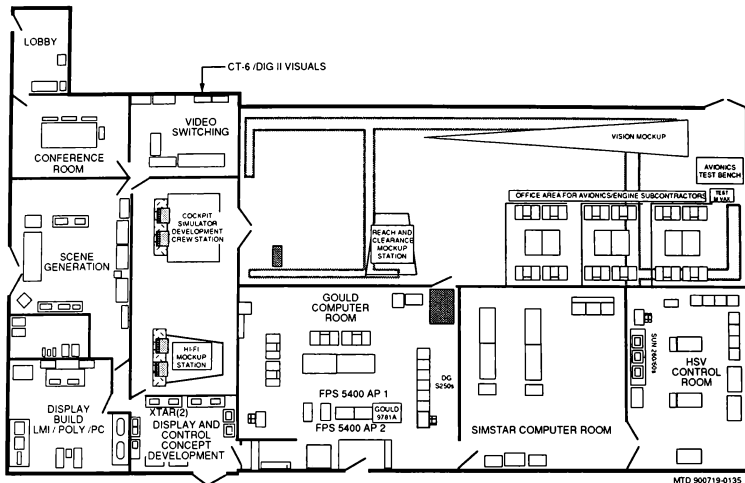


Fig. 2. The secure laboratory area dedicated to HSV development is a self-contained facility. This hybrid (analog/digital) engineering facility includes the capability for development and test as well as data collection/display.

- Analysis, operational technologies, demonstrations, and direct support of flight test definitions
- Crew station design
- Evaluation of flying and handling qualities
- Subsystems integration
- Failure mode analysis
- Instrumentation requirements and definitions
- Flight test techniques and procedure demonstrations
- Simulator sizing and performance requirements definition
- Crew systems design
- GN&C/engine/airframe integration
- Flight test analysis
- High-fidelity math models

HSV Simulator Development

The simulator development was divided into the following major tasks:

- Requirements generation for all software and simulator hardware design, including fabrication and systems operability
- Simulation math model/test case development and operations procedures
- Simulation software development and test
- Simulation data base generation, data collection/display/recording

The project efforts established test requirements for various integrated vehicle management system (VMS) candidates (GN&C, airframes, propulsion), created representative math models of these tightly coupled systems, produced simulation programs and support validations of these systems (collected data from sponsor-produced test matrices), and supported the data analysis of the simulation activities. The simulation activities involved support from several areas, including:

- Vision (out-the-window) systems
- Glass cockpit generation
- Human factors (crew-VMS interface)
- Flying qualities

The HSV laboratory host computer system is a Gould 9780 dual processor system with dual Floating Point 5400 array processors (aero and engine tables) and dual Data General S250 (input/output front-ends).

The laboratory also includes various workstations and computers: five Suns (display generation/data display), ten personal computers (display and control generation/display), three Poly 2000Es (visual displays; a complete video switching network), two LMI 2+2s (real-time adaptive guidance), and two sections of analog/SIMSTAR computers (high-frequency controllers/roll-out/landing, gear/tire, and braking).

Engineering capabilities provided by the simulation laboratory include the following:

- Real-time optimal trajectory planning (new guidance concepts)
- Optimal flight control (automatic and manual)
- Thermal management
- Model mission-oriented tests (consumables)
- Flight test data collection and reduction for large systems

All candidate VMS subsystems were math modeled and coded to support this task.

Vision Systems

The out-the-window visual displays are generated by three dual channel Poly 2000s and one Evans & Sutherland (E&S) system. The E&S CT-6 computing system is shared by various programs. Simulation of these viewpoints is critical to the design validation of direct-view structures for HSVs. Minimizing these fields of view, and still providing safe landings, are the basic design criteria.

Glass Cockpit Generation

The Sun workstations' software tool, DV-Draw, is utilized to create candidate display pages. DV-Draw functions are also used to label various parts of the graphic pages (static background, dynamic icons or objects, and special functions). A conversion routine converts or translates the DV-Draw data into "C" subroutines. Code for each specific application is developed and incorporated into the generic graphic display engine programs, which drive the "C" subroutines. The specific application code is usually the setting of initial conditions, or input/output functions. The final step is to compile and link together the "C" generic graphics engine code and the subroutine source code currently targeted for IBM/AT host (Fig. 3).

Crew Systems

The laboratory includes three separate crew stations: a crew development station (Fig. 4) with touch-screen monitors allowing for rapid prototyping, an anthropomorphic mockup for crew system functionality analysis, and a high-fidelity crew station with simulated active controls (McFadden throttle, center and/or side stick,



Fig. 4. The crew development station, a foam-core mockup, allows for rapid cut-and-paste concept development. This station also doubles as a repeater display station of the high-fidelity mockup, providing a second instructor/engineering feedback site during production/test support.

rudder, and brake pedals) and three out-the-window view points. The fidelity of these mockups stresses engineering functionality and exclusively supports engineering analysis. The following are primary crew system functional test objectives stressed in the simulators:

- Mission planning/replanning analysis
 - Attitude
 - Mach
 - Flight duration
- Air crew station geometry
 - Anthropometry
 - Reach
 - Vision
- Task allocation
 - Optimized crew performance
 - Automation

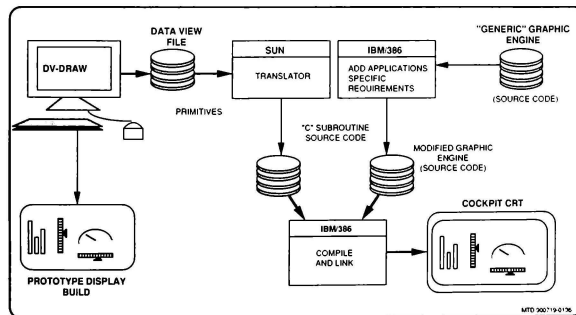


Fig. 3. The design and creation of "glass cockpit" displays is accomplished through the use of COTS software and in-house-developed software. Rapid prototyping and automation of display concepts produce cost-effective multifunctional displays, enabling crew station (human factors) evaluations of candidate VMS subsystems data.

Former astronaut Joe Engle is shown in the anthropomorphic mockup (Fig. 5) during pressurized and unpressurized reach-and-touch testing. Engle, having had the unique experiences of flying both the X-15 and the Shuttle, provides valued analysis of all crew station functionality.

The high-fidelity mockup (Fig. 6) is used for real-time pilot analysis/assessment of proposed crew system concepts. Undesirable and deleterious control characteristics are eliminated early in the design phase. The simulator models are coded to facilitate rapid cycling of flight conditions. As many as 70 runs per hour have been flown, allowing the results to be incorporated quickly into the design process. The simulator's mode control can be run in either a master (simulator-controlled) or in a slave (pilot-controlled) mode. This option allows the pilot to control and rapidly run specific handling-qualities tests without simulator operator intervention.

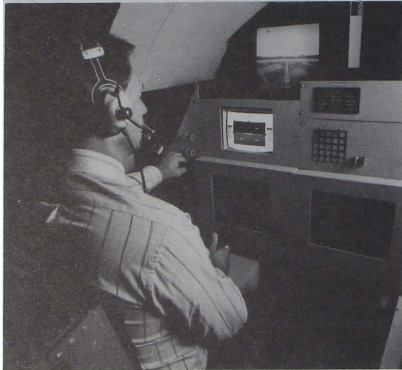


Fig. 5. The anthropomorphic mockup supports cockpit evaluations of visibility (both external and internal), and ingress/egress.



Fig. 6. The high-fidelity mockup is utilized to support crew station design (vision, geometry, vehicle control, subsystem control, etc.) and supports all major evaluations of flight handling qualities. The manned simulations enable Rockwell to investigate potential designs very early in the conceptual design phases and ensure performance objectives in simulated flight test.

Very Tightly Coupled GN&C/Engine/Airframes

The simulator is a major forcing function for subsystem integration, especially when testing man-in-the-loop operations. The math models are built to permit rapid implementation of different sets of programs and data.

The HSV airframe models utilize separate engine-on and engine-off data bases. Corresponding GN&C gains and functions also change with these same aero variations; plus, additional changes occur because of engine-on stages (low speed/RAM/SCRAM). Additional complexity is added when the guidance/thermal/engine controller feedbacks are considered. Separately decoupled system simulations are no longer possible. Minimum computational frequencies are 100 Hz for environment; 50 Hz for flight control; and various lower frequencies for flight guidance, thermal, and such systems as displays/visuals.

Flight Test Descriptions

Mission test phases are identified in Table 1. The test phases correlate with operational and flight envelopes (Fig. 7) for the phases identified in Table 1. The X-30 specific test objectives include analysis and evaluation of

- Flying and handling qualities
- Crew systems
- Thermal management
- Subsystems simulation
- Airframe analysis
- Propulsion system analysis
- GN&C analysis
- Attitude control
- Fuel/center-of-gravity management

The simulation is used to evaluate these concepts well ahead of future contract requirements to pave the way for the next phase in the HSV development cycle. Critical flight profiles include the following:

- Takeoffs (weight/center of gravity/thrust variation)
- Powered landings (weight/center of gravity/thrust variations)
- Level flight
- Acceleration and orbital operations
- Unpowered entry/landing

The simulation test plan is designed to explore HSV technology in a way that will expand the vehicle flight envelope from subsonic to extremely high Mach numbers. The simulation test plan parallels the HSV proposed flight test plan and flight test objectives.

Objectives established in the simulation test plan are expanded as capabilities are added to the simulator. This arises from the role simulation plays as a subsystem test validator and test tool.

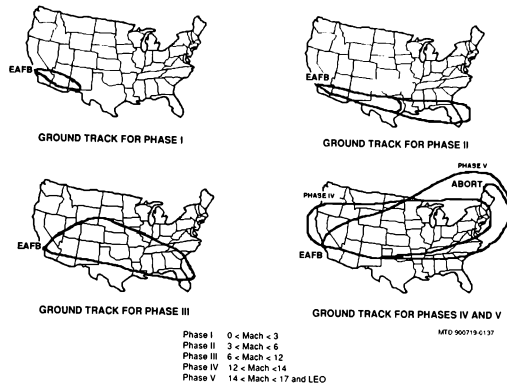


Fig. 7. The flight test envelope is broken up into regions of engine and aircraft performance. Regions of performance are correlated with primary test mission subsegments. All simulation tests are then related directly to these test objectives and flight phases.

Table 1. HSV Simulation Test Objectives for Flight Mission Phases

Flight Phase		Demo Objectives
Phase 0		HSV model implementation
Simulation site validation		
Phase 1	Mach < 3	Unassisted takeoff and landing
Envelopes		Powered go-around at landing
		Ferry capability
		Engine-out/restart
Phase 2	Mach < 6	Acceleration profiles
Envelopes		Ferry capability
		Engine-out/restart
Phase 3	Mach < 12	Hypersonic cruise
Envelopes		Loiter
		Engine-out/restart
Phase 4	Mach < 14	Hypersonic cruise
Envelopes		Loiter
		Cross range
		Engine-out/restart
Phase 5	Mach < 17+	Hypersonic cruise
Envelopes		Cross range
		Single stage to orbit
		Engine-out/restart

High-Fidelity Math Model

The simulation math models include a complete real-time 6-degrees-of-freedom configuration. The vehicle is flown over an oblate rotating earth with a nonspherical ($w/J2$) gravity model. The vehicle model has variable mass and center of gravity. The rotational model is described with both Quaternion and Euler equations. The translational motion is solved in terms of inertial coordinates. At least nine separate reference frames are maintained in real time (inertial, earth, body, platform, navigation, landing field, vehicle structural, wind, and stability). Fig. 8 shows the overall HSV math model and the signal paths between mockups and the subsystem models. The modularity of this system allows for model substitution with hardware replacement as it becomes available.

Conclusions

Rockwell has produced a test bed simulator capable of simulating a complete single-stage-to-orbit HSV. The operational flight

areas have also been expanded from FY 1988 capabilities to include unpowered deorbit, entry, approach, and landing, and powered takeoff, go-around, ferry, and hypersonic cruise. The subsystem accomplishments include an integrated high-fidelity crew station (split throttles, control sticks, various mode control, and meter data displays), enhanced out-the-window visual systems (various occulted view points), and audible feedbacks (tire and main engine sound effects). The simulated flight test areas ranged from subsonic (Mach less than 1) through hypersonic (Mach less than 27) including deorbit, insertion, and on-orbit. Initial specifications for generic HSV facilities and laboratories (VMS integration, training, and vehicle integration) were generated. New enhanced simulator requirements and corresponding data generation and display capabilities were produced.

Results and key findings are as follows:

- We identified and refined the requirements document for the simulator needed for HSV testing. These changes involve use of the high-fidelity crew station mockup, operations aspects of related testing, and all supporting software changes (new vehicle engine and aero combinations, hinge moments, on-orbit aero and gravity gradient, on-orbit digital auto pilot, flight controls, integration of thermal prediction algorithms, and various three-dimensional visual and display data bases).
- We generated all required math models according to the requirement generation refinements. Specific changes included new flight control (priority surface commands, new speed brake, and body flap); flex aero; hinge moment limiting; fuel accounting (attitude control system [ACS], main and auxiliary engines); variable centers of gravity; new atmospheres; various cockpit signals, switches, and displays; guidance updates (mixed FORTRAN and Ada), and various real-time operating system changes (interrupts, input/output, and hybrid analog and digital model distributions).
- We generated all necessary real-time simulation software (SW). These SW products include new baselined systems. The efforts concentrated on examining various VMS candidate subsystems and created representative SW modules of these tightly coupled systems, including the validation of

- Minimum landing and takeoff forward vision angles (based on minimum drag)
- New high Mach flight control laws, including major mode transitions
- More compact simulator designs and definitions based on trades of current simulators
- Latest global reference atmosphere model implementation
- Minimum visual transport delays based on handling qualities requirements
- Interface control documents (ICDs) for incorporation of on-board avionics test systems (interfaces to candidate tri-redundant general-purpose computer, bus system, and sub-systems)

The methodology incorporated in producing and implementing applicable data bases (flex and rigid aerodynamics, engine, hinge moments, flight control gains, etc.) was improved with emphasis on reduced speed and turnaround times of the simulation results. New, more standardized data tapes were used across various off-line performance simulators making comparisons of data faster and less susceptible to error. These results will be used to reduce all future development times and costs (lines of code, execution times, frequency of calculations, memory requirements, and input and output interfaces).

IR&D programs in the area of integrated vehicle control (thermal, guidance, engine, and flight) and crew systems design are developing control and vision requirements necessary for HSV-piloted flight. Results of these studies have impacted favorably on both the controls design as well as the cockpit layout and presentation, including definitions of out-the-window fields of view.

Future Activities

The HSV laboratory will continue to support NASP, Shuttle II, and ACRV/AMLS vehicle development; VMS laboratory; and related subsystem definitions such as the inertial measurement unit/air data system and flying qualities/controls definitions.

Acknowledgement

This work was accomplished under Rockwell's Space Systems Division's IR&D projects—193/299 (February 1988 through September 1990).

References

1. B.W. Mount and J. L. Simmons. Hypersonic Vehicle Real Time Simulation Requirements. Rockwell International, North American Aircraft Operations, NA-88-1416-L (October 1, 1988).
2. Aircraft Definition Statement (ADS) for the D791-469, Rocketdyne TMR Status Configurations. NASP Airframe Technology Program NA-89-68 (March 21, 1989). (Classified)
3. W.G. Burnett Final Report for FY 1989 IR&D Project 89193: Hypersonic Vehicle Management System (VMS) Simulation. Rockwell International, Space Transportation Systems Division, IL 298-130-SST-89-005 (September 18, 1989).
4. J.L. Simmons and W. G. Burnett, Hypersonic Vehicle Real Time Simulation Requirements. Rockwell International, Space Transportation Systems Division, IL 298-130-SST-89-006, Revision B (October 1, 1989).

David M. Draffin
Software Supervisor
John S. Bacha
Laboratory Systems Lead Engineer
The Integrated Technology Development Laboratories
Boeing Military Airplanes
Seattle, Washington

Abstract

The Boeing Company has built a secure, multi-project facility named the Integrated Technology Development Laboratories (ITDL).

This paper will give some background information on how the ITDL came into existence, along with methods developed to provide for security and laboratory operations. Some details will be discussed on the types of resources that are shared between projects, the software design, the hardware design, the power and ground, the communications system, and the configuration management needed to support secure, rapid configuration changes. The facility organization and logistics are also outlined. The paper concludes with some of the challenges experienced in executing the concept of a secure, multi-project facility.

The ITDL staff focuses on improving multiple, rapidly changing, secure, simulation environments. Current simulation capabilities of the ITDL are similar in nature to other facilities supporting air combat simulations. Because of these similarities, facility simulation capability will only be described when it relates to obtaining the goal of rapid configuration changes in a secure environment.

Introduction

Engineers and scientists continually investigate technologies and methods which will increase aircraft system performance and mission effectiveness. This has resulted in military aircraft becoming very complex. Traditional aircraft system laboratories, however, have been ill-equipped to support the research and development of such complex systems. Laboratories operated autonomously. Various aircraft subsystem research and design was performed on incompatible systems located in separate facilities. It was mainly due to technology limitations that systems could not be easily tied together, thus reducing the scope of integration and testing.

The last few years, however, have given rise to tremendous enhancements in computer technology, development of high-

speed local area networks, improvements in digital scene generators, and creation of low-cost, large scale, tempest shielding techniques. To take advantage of these improvements in technology, Boeing decided to build a single laboratory which could provide engineers with an environment where they could study highly integrated aircraft systems. The studies could start at the research phase, continue through demonstration and validation phases, and finish at the initial full-scale development phase. Realization of a Boeing commitment to build such a laboratory became evident in 1984 when construction of the Integrated Technology Development Laboratories began.

The ITDL grew from Boeing's Visual Flight Simulation laboratory. The Visual Flight Simulation organization was a relatively small flight simulation organization consisting of fourteen staff members and six laboratories. Visuals were generated with model boards or a 70 millimeter film projector. Only one or two projects could effectively be supported concurrently.

Substantial changes took place as the Visual Flight Simulation grew to become the ITDL. Methods were developed to support the security needs of projects, and the associated configuration management requirements. The infrastructure of the organization changed to accommodate growth, a substantial increase in the quantity of systems supported, and expanded responsibilities.

The changes to the organization occurred quickly. The facility was ready for occupation in November of 1986 and the entire contents of the previous labs were moved into the ITDL by March of 1987. The ITDL performed its first dual dome demonstration using a Compuscene IV visual generator in October of 1987. By this time simulation capabilities reached levels similar in nature to those described in the referenced Northrop^[1] and Lockheed^[2] papers.

As projects began moving into the ITDL near the end of 1987, the staff organization expanded to over 100 personnel to support them. With the influx of secure projects, one of the first challenges the organization faced was developing security procedures.

Security

Procedures were needed that could meet project security needs, yet accommodate the multi-user facility concept. There were few examples of how several projects, running at a variety of classification levels, could simultaneously exist in the same building. ITDL security levels were to range from unclassified through Top Secret.

The ITDL staff and the Boeing security organizations developed security procedures to accommodate both project requirements and ITDL goals. Methods and procedures continue to evolve as technology advances and new projects are supported. The methods involve three aspects of security: physical security, tempest security, and automated information system (AIS) security.

Physical Security

Physical security refers to the physical safeguards used to prevent unauthorized persons from gaining access to classified information⁽³⁾. The ITDL implements several physical safeguards.

The perimeter walls of each laboratory extend from the true floor to the true ceiling. The solid-core doors on the perimeter walls of each laboratory have built-in combination locks, cipher locks, and door alarms. Additionally, the laboratories are equipped with motion detectors, and the air-conditioning ducts are alarmed. These physical safeguards prevent unauthorized access to each laboratory and provide a warning if an intrusion occurs. Perimeter walls and doors of the ITDL facility are constructed in a similar fashion and are equally equipped with alarms.

Physical security also controls the communications between laboratories. The fiber optic cables are housed in sealed conduits installed in the hallways. Daily visual inspections of these conduits are made to assure no tampering has occurred. Fiber optics modems and patching control reside in each laboratory, and are therefore secured when the laboratory is secured.

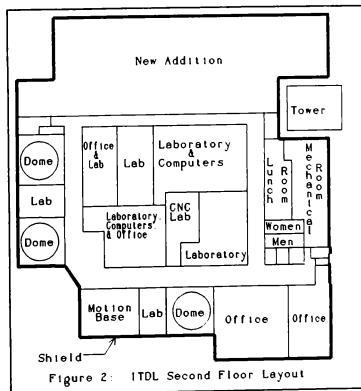
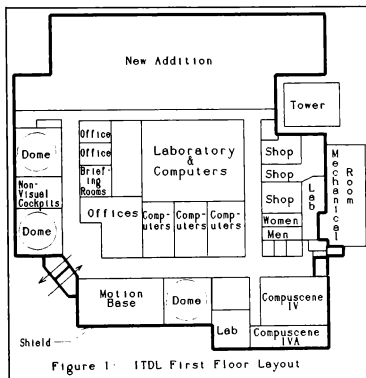
Tempest Security

Tempest security refers to the measures taken to prevent compromising emanations from systems used to process classified information⁽⁴⁾. Compromising emanations are unintentional signals that can, if intercepted and analyzed, disclose classified information. Facility shielding, equipment placement, power, grounding, and fiber optics are all utilized in the ITDL to reduce emanations.

The entire 104,000 square feet of the building is shielded. The shielding reduces building emanations more than sixty decibels within the ten kilohertz to one gigahertz range. The shielding

is applied to the exterior surfaces, i.e., roof, outside walls and bottom slab, rather than the conventional application on the inside room surfaces. Anterooms with twin tandem tempest doors are used to retain complete shielding during personnel ingress and egress. The shielding process is relatively inexpensive and provides more flexibility than individually shielded inside room surfaces. Figure One and Figure Two show the laboratory layouts, the shield and the facility ingress/egress anteroom.

Building emanations are reduced additionally by using specific interior construction materials, and by placing electronic equipment a minimum of three feet from the perimeter walls. In addition, the telephone service and all the alarm systems are filtered.



Power distribution and control also contribute to the tempest strategy in the ITDL. An isolated power system is provided for each laboratory and it is controlled to ensure that each power system services only the laboratory assigned. Each laboratory is also equipped with its own power transformer and power cart. The transformers provide one megohm of isolation between the building steel and power transformer inputs.

A single point ground is an essential requirement in the ITDL. The building structure, including the shield envelope and all systems, is grounded to a single point with less than five ohms impedance. Laboratories are not allowed to connect together using wire. This provides absolute physical control of all laboratory equipment connections.

Emanations are further reduced by use of fiber optic cables. Since fiber optic cables do not emanate, they are the only communication path allowed between secure laboratories.

Automated Information Systems Security
Automated Information Systems security refers to the requirements and procedures necessary to protect classified material processed or resident in computing systems from inadvertent or deliberate compromise. Automated Information Systems security requirements are the same in the ITDL as in other secure Boeing facilities, but because ITDL laboratories routinely change classification levels several times a week, classification and declassification procedures are different. Each project that schedules laboratories adopts a set of Standard Procedure Practices which adhere to the project security requirements. At the end of the secure session, every system in the scheduled laboratories are declassified according to those established procedures. Removable disk packs and removable hard drives are standard on the ITDL securable computers to allow the computers to be declassified without destroying the classified application software and generated data.

Resources

Each major resource in the ITDL is housed in an independent laboratory. These laboratories may be dedicated to projects or may be shared among projects. This mixture of dedicated and shared laboratories gives small projects access to affordable simulations, yet gives larger projects the flexibility, power, and local control needed to meet their goals.

Shared

The shared laboratories are available to any project on a scheduled basis.

Shared computer assets include three host simulation computer systems each consisting of a Gould 97/80, a Gould 97/50, and a Star array processor. Two laboratories contain General Electric Compuscene IV computer image generation systems for a capability of driving three domes. Another shared laboratory houses a Gould 97/80 development computer and a Vax 8700 development computer. Other shared assets include three 30' domes, a sensor tower, electronic shops, a machine shop, a video laboratory, three tactical fighter cockpits, a side-by-side cockpit, low fidelity crewstations, and various graphics computers for driving cockpit displays.

Small projects cannot afford dedicated laboratories, however, they can request ITDL resources to be scheduled and configured to meet their needs. Existing staff skills can also be used by the projects to prepare hardware and software for the project test bed. These skills and resources allow the principal investigators to focus their expertise, time, and money on the specific area of study rather than the infrastructure and mechanics needed to support and build the tests.

Larger projects use the shared resources for large simulations. For example, a project can develop and evaluate a new avionics suite using requirements derived through AASPEM⁵ (Air-to-Air System Performance Evaluation Model) analysis. This initial analysis is conducted in the project laboratories. The ITDL staff then supports the project in the integration of the system with flight control systems in other laboratories. At any stage, a project engineer may decide to "fly" the system on appropriate simulators to evaluate the design's impact on a mission. These additional resources are simply scheduled and connected using the appropriate fiber optic networks. This integration of shared resources gives large projects the flexibility of conducting comprehensive pilot-in-the-loop evaluations early in the development life cycle.

Dedicated

Projects can also request dedicated laboratory space. The grounding system, power distribution system, and fiber optic communications make it possible for projects to modify their laboratory without impacting other ITDL projects. Managers of the project have wide latitude in how their laboratory space is utilized, and in the computing equipment installed. This local control can be critical for projects responding to rapid changes in project requirements. ITDL staff retains responsibility for connecting dedicated laboratories to the power carts and to connect the inter-laboratory, fiber optic, communications system.

Most of the Boeing Military Airplane technology groups have dedicated laboratories and are permanent residents of the ITDL. These include groups which develop flight controls, avionics, weapons integration, photonics, propulsion, radar, electrical power, and crewstation design. Again, these laboratories are under control of their local management. The location of these technology groups within a single building substantially improves communication across discipline lines.

Dedicated project resources eventually must connect with each other or with shared resources. This, however, does not force the ITDL or its projects to standardize on any specific set of computing or test equipment. Instead, the communications systems have been selected which interface to a wide variety of vendor equipment. Interface control documents were written to define the interfaces and specific protocols established for data transfer compatibility. Typical simulations run a mixture of equipment from vendors such as Gould, DEC, Ironics, Evans & Sutherland, Silicon Graphics, and General Electric. The ITDL communications system acts as like an equalizer between different functional units of the simulation.

Rapid configuration

Projects will not benefit from sharing resources if the resources are constantly in use by other projects or being reconfigured. To utilize the capabilities of the ITDL systems, methods have been developed which reduce the time it takes to make configuration changes. These include software, hardware, communications, configuration management, power, electrical ground, and simulation architecture.

Software

Software designs, structured analysis, and compatible operating system software aid in reducing the time it takes to configure shared resources. Operating systems and computer libraries on multiple systems are maintained at the same version level, and in general are all updated at the same time. Programmers are required to develop all software using the computer libraries. If new library modules are needed, they are added to every system to assure systems compatibility.

The ITDL staff use structured analysis and design techniques to develop modular software. Modular software reduces development, debug, and integration time. When a project has a specific requirement that cannot be fulfilled by an existing package, a module within the package can be replaced by a project

specific one. If the project has hardware that is simulated using a software module, that module can be replaced with a hardware device. Using modules eliminates the need to redesign entire software packages. Further, all ITDL simulation packages involving a single vehicle (or participant) contain the software modules necessary to permit connection to a multiple participant simulation.

This approach to software design has reduced the number of problems inherent in continually changing resource configurations. Less software is modified, thus limiting the scope of problem solving when moving software between computers. Quickly adding or deleting modules in a software package simplifies the transition from one simulation to another, especially when large and complex simulations are configured.

Hardware

The ITDL has twenty-eight laboratories which contain hardware resources. Like the software, the hardware resources are modular in design. Modularity decreases development time and down time when modifications are needed. In addition, several shared resources are configured alike. The configuration of the three domes are nearly identical. The forward section of the tandem cockpit, and three other tactical cockpits contain the same basic capabilities. Three simulation computers are identical and are part-compatible with the development computer. The list of other, multiple, compatible resources goes on. Similar, compatible, configurations mean that if any one system is out of service, or scheduled with another project, an alternate system can easily take its place. The modular approach, again, increases the effectiveness of the ITDL.

Communications

For the modular approach to work at all, the resources in each laboratory must be connected together. Without a flexible and reliable communications system to do this, the costs of resource sharing would outweigh the benefits.

The Communications Network Control (CNC) System provides all laboratory communications. Utilizing over 215 miles of fiber optics the CNC supports multiplexed RS-232, Ethernet, ProNET 10, ProNET 80, multiplexed audio, and video. Equipment racks, called CNC bays, in every laboratory contain communications network control equipment such as patch panels, modems, multiplexors, and multi-port ethernet transceivers. Fibers leave the CNC bay in a secure, sealed conduit and connect directly into the central fiber optic patching room called the CNC Laboratory. Here, two patching methods are used to configure a network: automated and manual.

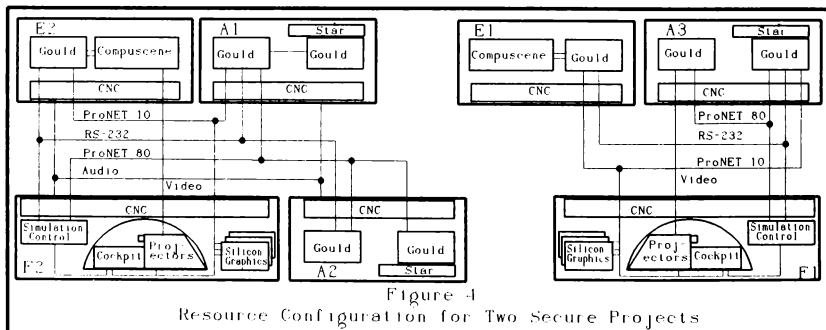
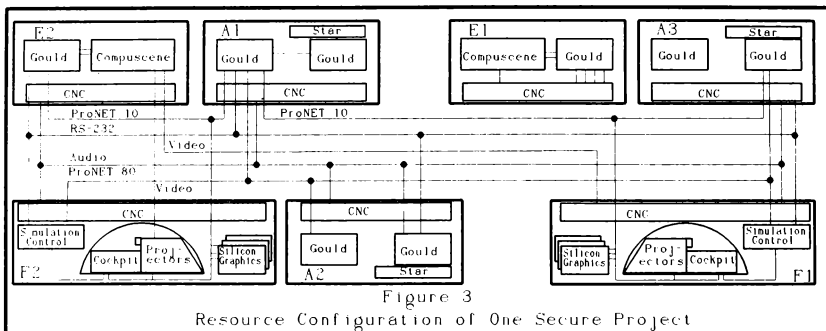
Passive fiber optic switchers controlled by a MicroVax are used to automate patching. Using a 'patch list' stored on disk, the MicroVax instructs the switchers to change the patch configurations. Each of the six switchers mates up to eighty-four fibers and can make forty-two separate connections at the average rate of one-half second per connection. Computer-controlled patching is utilized because it more accurate than hand patching and is much faster. Since the passive switchers do not convert light pulses to electrical signals the number of modems is reduced, reliability increased, and troubleshooting time decreased.

Manual patching is used for all classified processing. The CNC Laboratory is composed of twenty-eight laboratory lockers and seven project lockers. The project lockers allow secure patching between the twenty-eight lab lockers. The CNC bay of each laboratory is connected to its own lab locker via the sealed conduit. Networks are manually patched in these lockers. Once patched, each lab locker used in the project configuration and associated project lock-

ers is locked down with project-provided locks. The lockers are considered secure because once locked down, the fibers are completely enclosed in steel from one laboratory to another. Up to seven secure networks can be configured simultaneously in this manner.

Figure Three and Figure Four show typical ITDL simulation configurations. Figure Three represents a secure, dual-dome scenario. Resources in Labs A1, E2, and F2 simulate the blue fighter participant such as an F-15. Resources in Labs A3, E2, and F1 simulate the red fighter participant. The computer in Lab A2 generates blue and red wingmen participants, and missiles. The computer image generation system in Lab E2 creates the out-the-window visuals for both domes.

Within two hours, the configuration in Figure Three is changed to the configurations in Figure Four. This includes manually unpatching and declassifying the systems in Figure Three, and then manually patching and classifying the systems for the secure configuration presented in Figure Four.



In Figure Four, the resources in Lab A1, F2, E2 may be supporting a manned blue participant. The computer in Lab A2 may be simulating several blue wingman, several red opponents, and missiles. The scenario in Labs A3, F1, and E1 may simply be a flight controls test involving no other participants. The goal time for a configuration change of this size is one hour.

Configuration Management

Configuration management is critical in a complex and constantly changing environment where multiple projects utilize the same laboratory. A system was developed that meets the needs of security, configuration control, and projects while minimizing overhead and time delays.

The documentation and control of the standard configuration is called a baseline. A Standard Procedures Practice (SPP) document describes the baseline equipment, security practices, equipment classification procedures, and equipment declassification procedures.

Configuration changes are divided into 2 types: short term and long term. Short term changes are defined as temporary changes of less than 6 months of duration. Long term changes exceed the length allowed for short term changes and typically become permanent.

Implementation of a short term change is done by red-lining the current baseline and SPP documentation, obtaining security approval, and noting the change in the laboratory notebook. The manager signing off the change is responsible for assessing the changes impact, and communicating the impact to affected parties.

Long term changes to the baseline are made by generating a Configuration Change Request. The Configuration Change Request is reviewed by a change evaluation team comprised of ITDL staff representatives, Boeing facilities personnel, and project representatives. Implementation of approved changes involves obtaining security approval, updating the relevant baseline documentation and the modification of the SPP document.

Using this configuration management system, changes to any ITDL hardware and software system can be submitted, communicated to all necessary parties, approved, and implemented within a week. Depending upon circumstances, some changes can be approved and implemented in a day. Released documentation updates sometimes lag the change implementation, but at all times preliminary or red-lined versions of the documentation accurately reflect the system.

By implementing this system of Configuration Change Requests, SPP, short term changes, pre-approved equipment lists, and log books, we have been able to lighten the documentation burden of the project, maintain security, and still support a rapidly changing development environment.

Power

In addition to isolating power, power carts enhance the ability to change equipment configurations in each laboratory. Each cart has two power banks. Power runs are made in flexible metal conduit. To add new equipment, cable lengths, breaker sizes, and connector types are specified. The cable and breaker can be ordered and installed in the chaseway or under a raised floor in one day. Equipment and cables can be modified and connected without disrupting power to other equipment. Changes are thus quick, smooth, and efficient.

Electrical Ground

The grounding scheme also assists in configuration changes. Ground loops within a lab are eliminated because of the single point ground. Since all laboratories are required to interconnect using fiber optics, ground loops between labs cannot occur.

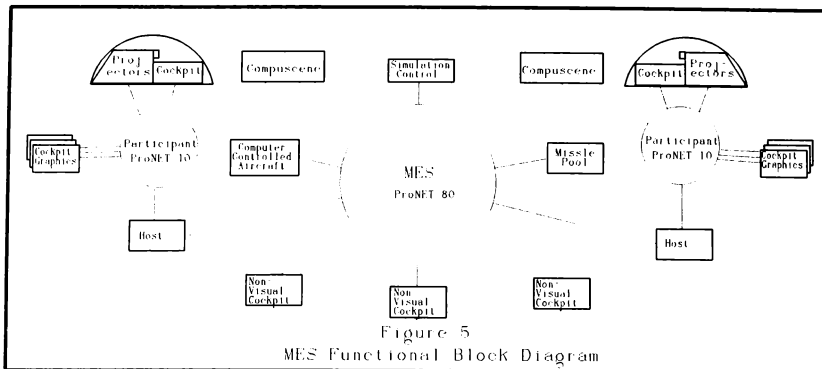
Simulation Architecture

Simulation resources are best utilized when the reconfiguration time is minimized. A reconfiguration time of under one hour has been chosen as a goal. To reduce reconfiguration time the simulation architecture is designed to support configuration changes, to be easily modified, and to be able to be scaled in size.

The architecture of ITDL simulations is composed of software and hardware building blocks. There are two levels of simulation building blocks: system components and participants. Multiple system components can be joined together to create a participant, and multiple participants are combined together as part of a Multiple Engagement Simulation (MES). This can be seen in Figure Five.

Typical examples of system components would be an avionics suite simulation, a flight controls system, an aerodynamics model, or a cockpit display set. These may be pure simulations or involve some level of actual hardware-in-the-loop. Integration of an appropriate set of these components yields a simulated vehicle which can be a participant in a combat simulation.

Several ITDL supplied participants currently exist. Multiple, computer-controlled participants such as mis-



siles, or aircraft, reside on any of the computer systems. The ITDL has three, hi-fidelity, multiple-computer, dome participants. Several lower fidelity, non-visual, piloted crewstations are also available. Each of these supplied participants have an interface that plugs into MES. Additionally, each participant has the capability to be enlarged or be reduced by changing system component modules.

The ability to scale the size of the simulations is important. A small-scale simulation, for example, requires a small number of resources. Reducing the number of resources lowers maintenance and security costs, decreases simulation complexity and reconfiguration time, and simplifies testing. This allows other projects to use the available resources. Expansion is easily accomplished by selecting the hardware, installing the disk packs, loading the participant simulation, and connecting the networks. Testing can be performed on a small number of resources and then easily configured to run in a full combat simulation.

Figure Five models the simplicity by which the various participants plug into the MES. It also shows the various components of the domed participants. The configurations of Figure Three and Figure Four were developed using this model. Participant connection to MES is as simple as shown in Figure Five because all required simulation data is placed onto the networks. This design and implementation has greatly enhanced the capabilities of the ITDL and yet has minimized the time it takes to configure simulations.

Organization and Logistics

Organization

The ITDL support staff is organized into five groups. These are Operations and Maintenance, Laboratory Systems, Laboratory Administrations, Software Systems, and Project Support.

Operations and Maintenance personnel maintain all shared equipment in the ITDL. A combination of in-house maintenance and contract maintenance is used in the ITDL. The maintenance approach used depends on the equipment, the vendor, the extent to which it is modified by the ITDL. Projects in dedicated laboratories may choose to perform their own maintenance or use the ITDL maintenance staff.

Laboratory Systems engineers perform all mechanical and electrical hardware designs, equipment procurement, and equipment installation. This includes the design of simulation cockpits, visual projection systems and special interface hardware. This group determines hardware system requirements and makes buy/build decisions. If the system can't be bought it is often built in the ITDL.

ITDL personnel in Laboratory Administrations are responsible for setting up and implementing configuration management procedures, and coordinating with projects for classified utilization of multi-use labs. The procedures set forth and enforced by this group are largely responsible for the ability of the ITDL hardware and software to function in a secure, multi-project environment.

Software Systems staff support and update the operating systems, develop project applications software, program the graphics generators, build software tools, and perform systems analysis. Software support even includes the negotiation of software site license agreements to cover all building residents. Software personnel designed and implemented the ITDL Multiple Engagement Simulation.

Project Support personnel coordinate all project needs with the ITDL staff and its management. Each external project such as contracts and Internal Research and Development, is assigned a Project Support Coordinator to function as the single focal point for that project in the ITDL. The Project Support Coordinator works with project personnel to detail laboratory and equipment resource requirements, the integration and demonstration schedules, and estimate ITDL staff hours required to successfully fulfill the project's requirements at the ITDL. In addition, the Project Support Coordinator is responsible for outlining the secure processing procedures, ITDL charging practices, configuration management, and other standard ITDL practices to the project. Project Support personnel, as a whole, are responsible for forecasting ITDL resource utilization and acquiring new business.

Scheduling of Shared Resources

It is common in many simulation centers for a laboratory to house multiple projects over a period of weeks or months. The ITDL is unique because ITDL laboratories commonly house two or three different projects in a single day.

Each week, schedule requests are submitted to the ITDL Project Support organization. Schedule requests are made by Project Support Coordinators representing the projects, ITDL staff personnel working on projects, or ITDL personnel working on ITDL projects. Requests must define the resource requirements, time blocks required and the configurations needed. These requests can include domes, computers, network configurations, and other shared or dedicated resources.

Sharing Personnel

In addition to sharing equipment resources, personnel are shared. A skilled pool of engineers, analysts, and technicians is available to be assigned to projects. As the needs of a project change, people with appropriate skills are moved on and off the project.

In most cases ITDL staff act as sub-contractors for a project. A project may need a cockpit modified, or a special I/O driver developed. The Project

Support Coordinator gives the requirements to the appropriate ITDL staff people. They become responsible for providing the defined functionality to the project. This includes meeting the project Interface Control Document requirements, documentation, schedule, and configuration management needs. This work typically takes place in ITDL staff laboratories rather than in project areas.

The teaming of a consistent laboratory staff with the various technology staffs and projects has proven quite effective in increasing communication across organizational lines.

Laboratory Allocation

Dedicated laboratory space is allocated in the ITDL based on several criteria. These criteria include security requirements, ITDL resource requirements, availability of laboratory space to other ITDL projects, and project priority. These criteria, explained below, are used as guidelines rather than hard, fast, rules.

Because the ITDL is a secure, tempest, laboratory facility, floor space is considerably more expensive than non-tempest office or laboratory space available in nearby engineering buildings. Project security requirements and the need to utilize ITDL unique resources figure heavily into laboratory allocation decisions.

Many projects require ITDL unique resources. Multiple projects sharing large capital assets on a day-to-day basis increases asset utilization. This provides a substantial cost advantage over the dedicated approach. Projects are also relieved of the burden of acquiring, maintaining, and managing these assets.

It is common for several Boeing Military Airplane technology groups to be working on different aspects of the same project. The ITDL provides the means of integrating these independent efforts. One of the prime goals in building the ITDL was to foster this type of multiple staff integration. This integration ability is typically not available in other facilities.

Independence

The ITDL infrastructure was developed to help minimize the turmoil normally associated with rapid changes. In the same way that each dedicated laboratory is managed independently of the ITDL management, the ITDL itself is managed independently from other organizations. This independence gives the ITDL manager the freedom to make decisions locally for shop support, hardware design, maintenance, and software design. This independence cannot be over-stressed. Time

critical contract work simply can not afford the luxury of waiting for outside support.

The ITDL manager, for the large part, controls the ITDL budget, decides which projects to support, determines priorities, and decides which tools are needed, based on Boeing Defense and Space Group goals. This allows for responsive, knowledgeable decisions to be made.

Complete shops for metal machining, electronics build-up, wood-working, and computer repair have been established. These shops are staffed by skilled personnel who report to the ITDL management. This allows flexibility to meet the resident projects needs.

Further Challenges

Developing and implementing the concept of the Integrated Technology Development Laboratories has been a challenging endeavor. A secure facility of this nature had not previously been built. Many valuable lessons have been learned and are being applied to the 40,000 sq. foot ITDL expansion now under construction.

Security

ITDL staffs are working with the Boeing security organizations to identify methods of streamlining procedures without compromising security. For example, an enhanced design for the secure fiber optic patch panels will result in substantially reduced patch times while maintaining a secure system.

Building Ingress/Egress

There are roughly 200 personnel in the ITDL on an average day. The building expansion is expected to add another 50. While the current anteroom approach to maintaining the shield during ingress and egress is effective, it is somewhat slow and requires periodic maintenance. Alternate approaches are being investigated.

Resource Scheduling

Currently the resources are scheduled manually. Information such as equipment status, preventative maintenance schedules, and network conflicts must all be collected and correlated by hand. A software tool is being developed to automate the collection of scheduling information, and to generate the actual schedule.

Communications

The Proteon ProNET 80 network is the backbone of the ITDL simulations. In most local area networks, data throughput between the host computer and the network is less than the network data rate. The large simulations in the ITDL require that each computer have access to the full network bandwidth. Cur-

rently available commercial interface boards are unable to support this high data rate.

To remove this limitation, the ITDL designed and built a Gould/SELbus interface board capable of the full ProNET 80 network rate. A similar effort is now under way for the VME bus interface.

Another Facility

The idea of a shielded, multiple project facility is so effective for Boeing, that a five-floor, tempest, office facility was constructed. It now houses most of the non-laboratory personnel associated with projects located within the ITDL. As part of the new addition, communication links have been established between engineering resources within the office facility and computing resources within the ITDL. Inter-facility networking and its security ramifications are a new twist in the ITDL list of challenges.

Conclusions

The enhancements in computer technology, development of high-speed local area networks, improvements in digital scene generators, and creation of low cost tempest shielding techniques have been applied successfully in the ITDL. This secure, multi-project facility has reduced costs, improved integration tools, maintained security standards, and removed redundant simulation facilities. The Integrated Technology Development Laboratories is a facility which provides a fertile environment for engineers and scientists to investigate technologies and methods to increase aircraft system performance and mission effectiveness.

References

- [1]Luhn, Jamison. "Tactical Air Combat in a Real-Time Multiple-Engagement Simulation." AIAA Flight Simulation Technologies Conference, 1988, Publication 88-4601.
- [2]Smith, Dorsey B. and John J. Soderberg. "Real-Time Tactical Simulation for Weapon System Development." AIAA Flight Simulation Technologies Conference, 1989, Publication 89-3315-CP.
- [3]Defense Intelligence Agency, "Physical Security Standards for Sensitive Compartmented Information Systems", Manual 50-3, May 1980.
- [4]Department of Defense, "Industrial Security Manual for Safeguarding Classified Information", Document No. 522.22-S-1, March 1984.
- [5]Boeing, "AASPEM Analyst Manual", Boeing Document No. D180-28789-1 Rev A, Airforce contract No. F08635-83-C-0267.

IMPLEMENTATION OF A BLADE ELEMENT UH-60 HELICOPTER SIMULATION ON A PARALLEL COMPUTER ARCHITECTURE IN REAL-TIME

Bruce C. Moxon*
MasPar Computer Corporation
Sunnyvale, California

John A. Green**
Advanced Rotorcraft Technology, Inc.
Mountain View, California

Abstract

The increasing role of real-time simulation in the development of complex systems continues to stretch the capabilities of conventional computer systems. As the scope and complexity of these simulations have increased, so has the need for innovative simulation development and analysis tools and high performance computing platforms. In this paper we describe a project aimed at bringing these elements together in a unique approach to the development of simulations for helicopter handling analyses. The approach combines sophisticated engineering design and analysis tools, powerful, scalable parallel processing technology, and a low cost pilot's review station to bring together simulation design engineers, computer scientists, and pilots as part of an evolutionary design team. This interdisciplinary, team-based approach expedites simulation design and validation, and can off-load expensive simulation facilities, thus reducing both the time and cost involved in the simulation development process.

Introduction

Over the last few years, a number of trends have developed in the application of computer technology to real-time simulation. Perhaps the most significant trend is that the systems being modeled and simulated are themselves growing increasingly complex. Recent years have seen astronomical increases in the complexity associated with such systems as aircraft, spacecraft, and sensor and surveillance systems. This increase in complexity has made the development of prototypes and subsequent environment testing infeasible in many situations. As a result, engineers have been turning more and more to computer based simulation for designing and prototyping in a cost effective manner.

As systems become more complex, the simulations we must develop to model the real-world behavior also become more complex. Intricate subsystems capable of sensing subtle environmental conditions, and control systems able to better integrate a variety of information require greater complexity in our mathematical models of systems and their interactions. More advanced models yield higher fidelity simulations that accurately reflect real-world conditions.

Additionally, attempts to model interactions between systems and their environment, and between different systems place even greater requirements on the computers used for simulation. For example, advanced helicopter simulations, such as the UH-60 Blackhawk model described below, are being "flown" in air-to-air and air-to-ground combat scenarios to see how they might perform in battle situations. The introduction of multiple model and subsystem interactions expands the scope of a simulation, and greatly adds to the computational requirements.

In many cases, interdisciplinary approaches to computer simulation are evolving to meet the needs of complex system simulation. Simulation of an elaborate, multiple unit engagement model using several aircraft might include aspects of all of the following: aerodynamic analysis (for flying aircraft), structural analysis (shock forces on ground based units, and aircraft response to aerodynamic and acceleration induced forces), control system theory (for aircraft and weapons system guidance and control), image and signal processing (for sensor systems), and mission planning. The relationship between a number of simulation disciplines is illustrated in figure 1.

Hardware-in-the-loop and man-in-the-loop simulations

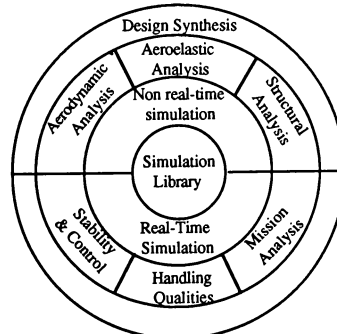


Figure 1. Interdisciplinary approach to simulation.

* Senior Applications Engineer, formerly Senior Applications Engineer BBN Advanced Computers Inc.

** Project Engineer, Member AIAA

have always had to respond to external and internal events in "real-time". Yet, as the real-world systems we attempt to model become more complex and more capable, the system dynamics also become faster, resulting in the need for faster simulation frame rates to guarantee simulation stability and integrity.

In response to the growing appetite for compute power, simulation engineers have been turning to more powerful computer technology. But even conventional technology is having a hard time keeping up with the computational appetite of advanced simulations. Parallel processing technology provides the computational capabilities required of such advanced simulations.

Figure 2 shows how the components of a generic flight simulator might be mapped across multiple processors to bring the simulation frame time under the real-time response barrier. Each of the major simulation subsystems might be distributed across one or more physical processors, combining both multiprocessing and parallel processing techniques to meet the real-time frame requirement.

While general parallel processing architectures can, in theory, provide the additional computational and I/O capabilities required by advanced simulation systems, they have thus far been applied to such problems with only moderate effectiveness. The additional capabilities afforded by such machines bring with them additional complexity -- in decomposing and restructuring applications to take advantage of parallelism, and in providing the necessary communication and synchronization between multiple processors. What is needed to harness the power of parallel

processing is a set of new simulation tools -- for analyzing problem structure, for implementing multiprocessing based applications, for verifying proper functionality, and for providing maximum performance.

In the following sections, we describe a simulation development and analysis platform that combines Advanced Rotorcraft Technology's (ART) parallel simulation development and analysis environment, *FLIGHTLAB*, with BBN Advanced Computers Inc.'s TC2000 Time Critical Computer System to provide a high performance platform for the development of real-time helicopter flight simulations.

We discuss the implementation of a Sikorsky UH-60 Blackhawk blade element simulation on this platform and compare the simulation performance characteristics to that of a uniprocessor-based approach. A brief discussion of future directions and possible implications of this work concludes the paper.

Background

For the past three years, ART has been working under contract to the Military Technology Office at NASA's Ames Research Center to investigate the application of parallel processing technology to advanced helicopter simulations. In the initial phase of this work, a simulation of the RSRA X-Wing¹ was implemented on a MicroVAX based system. A MicroVAX II was augmented by adding four Avalon AP20 auxiliary processor boards to the backplane, each of which had an Intel 80386 processor and a Weitek floating point co-processor. The parallel software architecture for the X-Wing² was designed to enable real-time performance on the MicroVAX based hardware. As a follow on to the X-Wing simulation, a UH-60 simulation was developed based on Sikorsky's GENHEL program. The basic four processor architecture was a simplified version of the X-Wing simulation, having the rotor distributed with one blade per processor.

Having shown the capability of a parallel architecture for real-time simulation, the next step is to improve the fidelity of the mathematical model, and to move the code to a more sophisticated parallel computer. The Military Technology Office and the Army Aviation System Command Aeroflightdynamics Directorate sponsored additional work to compare a rotor-map model against a blade-element model, and to investigate the use of an aeroelastic blade element model³.

The aeroelastic model is much more computationally intense than the rigid articulated model, and consequently requires a more powerful computer than the MicroVAX based system. In response to this task ART identified the BBN Advanced Computers Inc. TC2000 computer as being capable of running the aeroelastic model in real-time. The ultimate goal of this work was to drive the Crew Station Research and Development Facility (CSRDF) at NASA Ames with the aeroelastic model.

The first step towards this goal was to port the FlightLab software and the basic UH-60 BlackHawk

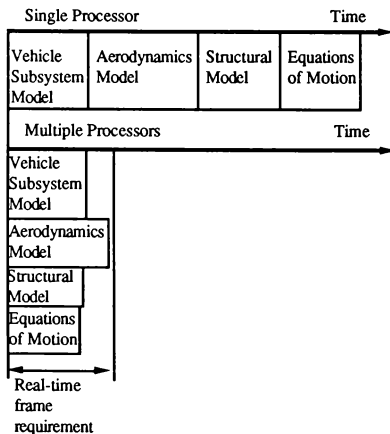


Figure 2. Simulation mapped across processors.

simulation from the MicroVAX, running VMS, to the TC-2000, running Unix. This involved modifications to the existing inter-process communication and synchronization model.

The configuration described here was demonstrated at the Inter Service/Industry Training Simulation Conference (ITSC) at Fort Worth, Texas, in November 1989.

A New Approach to Helicopter Simulation Design

Helicopter analysts have always faced a major limitation in the use of real time simulations for handling qualities analysis. The extreme complexity of rotorcraft mathematical models have made it impossible to accomplish real time simulation on all but the most costly computers. The traditional solution to this problem has been to reduce the complexity of the helicopter simulation by restricting the frequency range of validity to the bandwidth of the pilot's response. A higher bandwidth model is, however, necessary to satisfy the more stringent requirements of modern control systems and high frequency dynamics of the rotor system.

Blade element models of rotor dynamics provide the required bandwidth by representing the dynamics of each blade separately and partitioning them into segments for purposes of aerodynamic load computations. This high frequency updating of many blade segments is computationally intense and consequently requires a powerful computer.

NASA's Ames Research Center has been using a blade element rotorcraft simulation with its Vertical Motion Simulator (VMS) to ensure that the most accurate dynamic representation capable of real time operations is utilized in this sophisticated and costly motion base laboratory. The simulation software is based on a UH-60 Blackhawk blade element math model provided by Sikorsky from their

GENHEL simulation. In addition to the six degree of freedom rigid body modes, it includes blade flapping, lead-lag, uniform and first harmonic inflow, and rotor speed degrees of freedom.

Each main rotor blade has a flapping and lead-lag degree of freedom with five aerodynamic segments. The aerodynamic coefficients on each segment are computed from nonlinear tables as a function of Mach number and angle of attack. A uniform inflow is modeled by momentum theory. The first harmonic inflow is represented by Glauert terms as a function of wake skew angle. The drive train includes the clutch logic and a nonlinear T-700 turboshaft engine model with the electrical and hydraulic control systems.

The tail rotor is represented by a Bailey model with aerodynamic interaction. The aerodynamics of the fuselage/wing and tail surfaces are modeled with tabulated data obtained from wind tunnel tests.

Simulation Functional Decomposition

ART's interactive dependency analysis tool, *TRACE*, was used to reconfigure the serial UH-60 simulation to run as a parallel application. *TRACE* provides graphical data dependency and control flow information that is used to identify areas of the code for possible parallelization. This information is crucial to the creation of an efficient and correct parallel implementation.

The sequencing and data dependency of the simulation modules can be seen from the Gantt chart display in figure 3. The control modules, ROTCON (rotor controls), FPS (flight path stabilization system) and SAS (stability augmentation system) are first executed to generate deflections for the control surfaces. The ROTOR routine computes the aerodynamic loads for each of the four blades. The INTERFER module computes the aerodynamic interference effects of the rotor wash on the fuselage, tail, and tail rotor. The GEARBOX module sums the torques on the drive train for the rotor speed computations. Finally, FTOTAL sums all of the computed forces to drive the vehicle equations of motion, STRIKE.

Using the dependency analysis information from *TRACE*, coupled with an engineering analysis of the underlying simulation model, the program was reorganized for parallel operation on four processors. The four processor architecture was derived from the MicroVAX version of the simulation, and was retained to facilitate the porting effort in a short period of time. While this software architecture is more than sufficient to meet the frame requirements on the TC2000 system, it is perhaps a less than optimal mapping, as will be discussed below. The resulting configuration is shown in figure 4.

Since the ROTOR module is the most time consuming module, it is distributed over the four processors, using one processor for each blade. This is possible because the analysis shows the blades to be independent of each other. Similarly, the GEARBOX

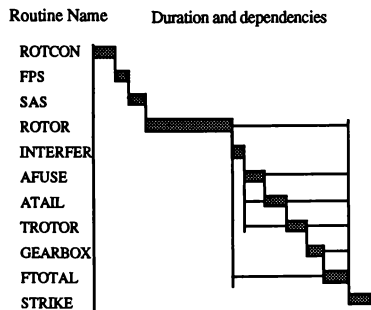


Figure 3. Sequencing and data dependency of the UH-60 simulation modules.

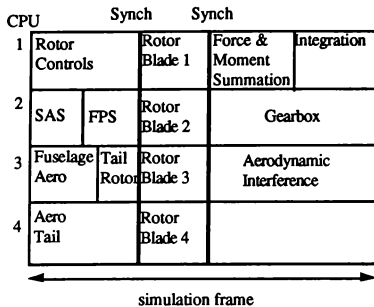


Figure 4. Simulation functional mapping to multiple processors.

routine can be coprocessed with the interference module, INTERFER, and the force summation and equation of motion routines, FTOTAL and STRIKE. These modules are all processed after the ROTOR module since they are dependent on rotor outputs. The other load generation modules (fuselage, tail, etc.) should also be processed after the control inputs and rotor interference effects. However, their dynamic response is so slow compared to the frame time, that they may utilize control and interference information that is one cycle old. This allows them to be coprocessed with the current control computations, resulting in a more uniform CPU loading and cost effective hardware configuration.

Interactive System Analysis

Once the simulation had been parallelized, it was supplied with an interface to ART's on-line,

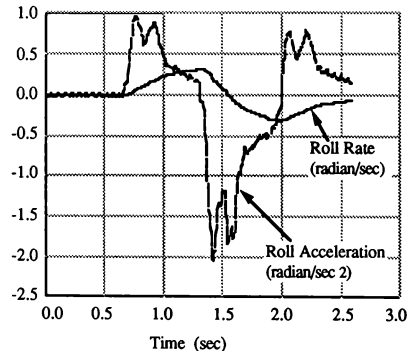


Figure 5. Simulation analysis using *SCOPE*.

multiprocessing executive, *SCOPE*. *SCOPE* is an interactive design analysis tool that supports on-line simulation analysis. Based on the public domain MATLAB matrix/vector manipulation package, with appropriate extensions to support interactive control and analysis, the package provides simulation engineers with a library of engineering analysis functions and graphical display utilities. Engineers can use *SCOPE*'s simulation interface to fully configure a simulation run, setting initial conditions and specifying time varying inputs, then execute the simulation and collect time histories of selected outputs. These outputs can be reviewed and analyzed using *SCOPE*'s utilities.

Figure 5 is a *SCOPE* plot showing the UH-60 Blackhawk's roll response to a simulated control input. Data from the simulation can be extracted and saved for later plotting or analysis – such as transfer functions, Bode plots, and parameter identification. It is also possible to generate plots such as fig. 5 in real-time.

Pilot Station Development

A "Pilot Station", consisting of an out-the-window display and representative pilot controls, allows the simulation designer to include the pilot in the development loop. This provides a more accurate and effective design methodology through rapid prototyping. It also gives the engineer a pilot's eye view of the simulation during development.

This real-time piloted capability adds a new dimension to integrated system design and analysis, allowing the pilot, or an engineer with piloting ability, to make a quick assessment of the effect of design changes and provide rapid feedback into the design process. It also expedites the checkout of real-time simulation by allowing the vehicle to be flown through a wide range of operating conditions in a short period of time and visually observing the response. This increased level of synergism between pilot and engineer will expedite both simulation validation and design, while off-loading expensive simulation facilities.

The pilot station uses a high speed graphics workstation to provide both "out-the-window" and instrument displays, and a set of pilot controls to provide a low cost "flight simulator". The Generic Visual System (GVS) package from Gemini Technology, provides out-the window displays, head-up displays, and cockpit instrumentation displays.

The TC2000 Time Critical Computer System

BBN Advanced Computers Inc.'s TC2000 Time Critical Computer System is a scalable multiprocessor computer. It employs a multistage switch interconnect to tightly couple up to 504 "function cards" in a single shared address space. Function cards provide varying amounts and types of processing, memory, and I/O capacities, allowing system configurations tailored to application needs. The TC/FPV function card uses a 20MHz Motorola 88100 processor, 88200 cache chips (16K instruction and 16K

data), 4 or 16 MB of memory, and an industry standard VMEbus interface. Each TC/FPV is capable of delivering 19 Dhrystone MIPS and 13.6 MWhets.

The TC2000's VMEbus interface has been tightly coupled into the function card. Through the use of two-way mapping RAMs, VME address space can be configured as part of the TC2000's system global address space, allowing I/O access to both the local processor on the function card and to all other processors in the system. Similarly, devices on the VMEbus can access both memory local to the TC/FPV card and, through the Butterfly switch, memory on other TC2000 function cards.

One of the unique features of the TC2000 is the notion of "clustering". Clustering provides a means of managing the processor, memory, and I/O resources provided by TC/FPV function cards on a system-wide basis. Using the clustering model, a number of applications can be run under a variety of execution environments and operating systems -- all on the same machine at the same time. Clusters can be used to manage specific hardware resources, such as the display and control interfaces in the helicopter simulation environment. Access to these resources can be granted or restricted with Unix "ugo"/permission modes. Clustering provides system developers with a natural way to think about partitioning their problems, allowing the allocation of specific resources to specific tasks.

The clustering concept extends to the TC2000's support of multiple execution environments through an open software architecture. Processors can be configured at boot time to run either the nX operating system (4.3 BSD Unix with a number of multiprocessing and parallel processing extensions), or the pSOS⁺^m real-time kernel. pSOS⁺^m is a true real-time operating system, with fast, deterministic context switch and interrupt latency times.

Memory can be shared between nX and pSOS⁺^m clusters, allowing applications to be built that span operating system boundaries. For example, the helicopter simulation in this application is targeted for execution in a pSOS⁺^m cluster, while the *FLIGHTLAB* simulation executive could remain nX based.

The TC2000 provides an X Window System based development environment, Xtra, that supports debugging, analysis, and performance tuning of multiprocessing applications. TotalView, a multi-window, multiprocess source language debugger, and Gist, an event based performance analyzer, were used extensively throughout the development effort.

Implementation

Pilot Station Hardware Interfaces

Figure 6 illustrates the hardware interfaces developed on the TC2000 to support the pilot's station. Data transfer between the Silicon Graphics display station and the TC2000 was provided by a DMA interface. This was implemented with two IKON 10089 VME/DR11 interface

boards: one in the Silicon Graphics VMEbus, and the other through one of the TC/FPV VME interfaces. The DR11 based interface was selected because of its superior performance over the GVS supported Ethernet interface, and for eventual integration requirements of the CSRDF.

The pilot controls were connected directly to the TC2000 using an analog to digital converter. The Data Translation 1401 A/D board that was selected provides up to 16 double-ended or 32 single-ended A/D channels. The current control panel employs 9 double ended channels, supporting a three-axis joystick (roll, pitch, and yaw controls), a single axis stick (collective), four on/off switches (master control, two SAS controls, and an FPS control), and a pushbutton (trim release).

Drivers for both the IKON board and the Data Translation board were developed under BBN Advanced Computers Inc.'s nX operating system.

Software Port

In this effort, the authors ported the *FLIGHTLAB* development environment, and a previously parallelized version of the Sikorsky UH-60 Blackhawk rigid blade element simulation to the TC2000. Device drivers for the DR11 and A/D interface hardware were developed to support the interactive *Pilot's station*, using a Silicon Graphics 4D/70G graphics display system and a custom control console. Additionally, DR11 DMA support was added for Gemini Technologies' Generic Visual System (GVS).

The effort involved the porting and creation of some 100,000 lines of source code, as shown in Table 1.

In the port of *FLIGHTLAB*, a number of significant modifications were made. First, VMS system calls were removed and, where appropriate, replaced with corresponding Unix system calls. Second, *FLIGHTLAB*'s runtime state vector symbol table facility was rewritten in C for improved performance, functionality, and maintainability. Third, support was added to *FLIGHTLAB*'s plotting facility to support output to the X Window

Description	Lines of source	Language / notes
<i>SCOPE</i> - simulation executive	20,000	Fortran (port) C (new)
<i>TRACE</i> - dependency analysis tool	45,000	C (port)
UH - 60 - simulation interface	3,000	Fortran (port) C (new)
Pilot's Station simulation interface	1,500	Fortran (new) C (new)
UH - 60 - blade element simulation	30,000	Fortran (port)
TC2000 memory management	1,000	C (new)
DR11 and A/D device drivers	2,000	C (new)

Table 1. Porting Effort.

System Tektronix 4014 terminal emulation mode.

New code was generated to support multiprocess operation on the TC2000. This included the process and memory management functions needed by the *FLIGHTLAB* executive, as well as interprocessor communication and synchronization protocols for interaction of the simulation processes with the executive and with each other. Here, the TC2000's process and memory management facilities, and shared memory atomic operations were used to build an efficient and reliable foundation for multiprocess execution.

Software Architecture

The basic software architecture consists of the *FLIGHTLAB* executive, the multiprocess UH-60 Blackhawk simulation, and display and control drivers. These elements are coupled through a shared memory interface that provides interprocess communication and synchronization.

The *FLIGHTLAB* executive is used to control and analyze various aspects of the simulation. It creates the necessary simulation processes, starts the display and control interface processes, and continues to operate relatively independently of the other system components. The DMA and A/D device interface processes manage the flow of data between the Pilots Station and the TC2000. Finally, the four simulation processes run concurrently synchronizing as necessary to retain the integrity of the simulation.

The system was partitioned along functional lines using the TC2000's clustering capabilities, as shown in table 2. Clusters were created at system startup to house each of the device interface processes. The function cards through which the two VME devices interfaced were selected as the physical "nodes" for these clusters. The clusters were given the symbolic names "DR" (for the node connected to the DR11 interface) and "DT" (the Data Translation interface) using the TC2000's cluster naming features. Additionally, a "SIM" cluster was created to house the 5 function cards that support the *FLIGHTLAB* executive and the simulation processes. These processes could have been split across two separate clusters, a single

processor cluster for the executive, and a four processor cluster for the simulation itself. However, the additional partitioning would not have provided further utility in process or memory management, so the processes were lumped together in a single cluster. A future implementation (e.g. one in which the simulation processes and display and control interface processes run under pSOS⁺_m, with *FLIGHTLAB* running under nX) might, however, want to take advantage of clustering in this manner.

After the decomposition was complete, the next task was to explore the three main software architectural issues of a multiprocessing application:

- Process management
- Memory management
- Interprocessor communication and synchronization

These are explored in more detail below.

Process Management

Both nX shell directives and nX system calls are used at program execution time to bind processes to the appropriate processors. The system call `getclusterbyname()` returns a single word `cluster_id` given a cluster name. This identifier is then used in the `fork_and_bind()` system call to create a process on a particular processor within the specified cluster. In the described configuration, the display interface process, `DISPLAY`, executes in the "DR" cluster, the control interface process, `STICK`, executes in the "DT" cluster, and the *FLIGHTLAB* and simulation processes each execute on a different node in the 5-processor "SIM" cluster. Hence each process is guaranteed exclusive access (except for nX system activity) to the processing and memory resources of its respective processor.

While the impact of nX system activity on these processes can be reduced to a minimum through the use of supported scheduler control, it can not be eliminated entirely. This is because nX is a truly parallel implementation of Unix, with distributed data structures, and parallel threads of control. All nX nodes contribute memory resources for use by the kernel, and hence minor interruptions in processing and/or memory access may

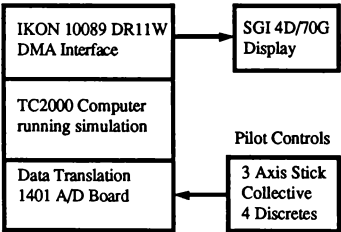


Figure 6. Hardware interface block diagram.

Cluster	Process	Description
DR	DISPLAY	Interface to SGI visual system
DT	STICK	Interface to pilot controls
SIM	FLIGHTLAB	Executive process
	Simulation Processes	Four simulation processes

Table 2. Cluster-based software architecture.

occur. For true real-time operation, these unpredictable system activities are unacceptable, and pSOS[™] would be the appropriate execution environment. However, due to schedule constraints, this system was developed under nX.

Memory Management

The simulation development environment employs the use of shared memory to support interprocessor communication and synchronization. At the time of implementation, the GLOBAL COMMON data attribute (providing common blocks that can be shared among unrelated processes) was not supported by the TC2000 FORTRAN compiler. Therefore, a prototype of global common was implemented using TC2000 FORTRAN ALLOCATABLE COMMON blocks. Storage was then provided at runtime via the nX memory mapping system call, `vm_mapmem()`. This provided all of the relevant processes with the necessary shared common block.

Upon execution of the Blackhawk simulation from within the *FLIGHTLAB* executive, two 8K byte pages of shared memory are "allocated" at the node where simulation processor 1 will execute. The first page contains the shared simulation state vector and a set of synchronization and protocol flags used to honor module dependencies within the simulation, and to coordinate communication between the executive and the simulation processes. The second page contains a buffer area used to pass commands and data back and forth between the executive and the simulation processes.

Simulation processor 1 was initially selected to house the shared memory because it most frequently accesses the shared state vector (particularly in summing the forces and moments and integrating the equations of motion) and because it is involved in the majority of transactions with the *FLIGHTLAB* executive. The placement and distribution of shared data is critical to optimal performance in the TC2000 architecture. The Butterfly switch supports concurrent memory references between disjoint sets of processors. Hence one gains increased memory bandwidth by distributing data among a set of memory modules and accessing the distributed elements concurrently from a number of processors. The TC2000 hardware architecture supports **hardware interleaving** -- distribution of contiguous physical address across a number of memory modules -- however, software support for this feature was not available in the version of the operating system with which we were working. For the initial prototype, we decided to bind all of the shared memory to simulation processor 1, and to explore the implications of that mapping and of hardware interleaving support through the TC2000's performance analysis tools.

Additionally, we decided to leave the shared memory supporting the state vector and IPC mechanisms as **non-cacheable**. This is the default cacheability attribute for nX shared memory. It provides for the safest and simplest use of shared memory under nX, as it makes software based

cache coherence protocols unnecessary. However, it has memory access performance implications in that cached "remote" (across the switch) references use an extended **cache line fill** operation to retrieve 16 bytes at a time across the switch, whereas uncached remote references retrieve a (4-byte) word at a time. Again, the goal here was to first port the simulation and later analyze architectural design issues for performance implications.

These two memory management issues, locality and cacheability, can significantly affect the performance of a parallel program. When developing high performance applications on parallel processing computers, it is in general best to implement a somewhat generic version of the code first, then to tune the code based on analyses of system performance. That has been the approach taken in this application.

The shared memory is set up by the executive through the use of the `vm_mapmem()` system call, nX's named shared memory facility. The executive then starts each of the simulation processes, as well as the two device interface processes, and waits for them to acknowledge existence through shared memory flags. Each of the simulation processes and the device interface processes attach to the same region of shared memory using the `vm_mapmem()` call, then set their respective acknowledgement flags.

Interprocessor Communication and Synchronization

The basic execution model for a frame based simulation consists of reading, updating, and writing the appropriate state variables from the simulation state vector each simulation frame. In a multiprocessing environment with a shared state vector, care must be taken to maintain any inherent system dependencies between simulation modules. Multiprocess simulations provide improved performance by distributing the computational workload of various simulated subsystems throughout the multiple processes, and hence require some sort of synchronization to guarantee the integrity of the implementation.

As has been discussed earlier, the basic simulation frame for the UH-60 Blackhawk simulation is subdivided into 3 simulation subframes. Each of the processes executes a different subset of the simulation subsystem models. The adherence to data dependency constraints is guaranteed by **barrier** synchronizations at the conclusion of each of the three subframes. During each subframe, the processes each read the appropriate state variables, execute the appropriate subsystem model(s), then write back the affected state variables. The dependency analysis performed using *TRACE* guarantees that each of the subsystems active in a given subframe operate on disjoint sets of "output" variables, again maintaining simulation integrity.

Debugging

After initial coding, some debugging of the modified communication and synchronization model was required. BBN's TotalView debugger was used to control concurrent execution of the *FLIGHTLAB* executive and simulation

processes, and to pinpoint otherwise subtle multiprocessing hazards. Errors in multiprocess communication and synchronization logic were effectively located by independently breakpointing and stepping the various processes involved in the transaction.

The functional accuracy of the simulation was next validated using *SCOPE* tools for simulation analysis. Using *SCOPE*'s offline simulation mode, synthetic control inputs were generated, and the response of the associated state variables was compared with those of the serial simulation.

Performance Analysis

As with any frame-based simulation, the predominant performance metric is the system frame time requirement. Hardware-in-the-loop systems have firm frame time requirements related to the speed at which various devices operate. Man-in-the-loop systems, including training simulators and handling qualities analysis simulators have frame time requirements dictated by the natural frequencies of the systems being modeled. These requirements are just as firm as those imposed by hardware devices, as failure to meet them jeopardizes the integrity of the simulation on the whole.

In a blade element rotorcraft simulation, the update rate of the blade aerodynamic loads is the critical computational element driving the simulation frame time. An accurate and numerically stable solution of the blade dynamic equations requires that they be updated every eight degrees of azimuth travel. At the rotor speed of the UH - 60, this translates to an eight millisecond frame time.

On a TC-2000 equipped with 16 MHz versions of the TC/FPV function cards, the simulation executed in a 4.6 ms average frame time. This is well under the 8 ms requirement of the simulation, leaving ample reserve to improve the fidelity of the rotor model, or to model additional subsystems while maintaining real-time response. This is one of the promises of parallel processing technology for real-time simulation.

It is important to note the presence of the word "average" in the performance figures quoted above, as it implies that some frames executed faster than the 4.6 ms average and some executed more slowly. This distribution of execution times is normal under a Unix operating system, due to varying levels of system activity, "softclock" hits, etc. Variance in frame execution time can be as much as 1 - 2 ms under nX . While this leaves us well within the 8 ms frame time requirement of the prototype, it provides unacceptable deviations for simulations with more aggressive frame time requirements.

While the 4.6 ms frame time is impressive in its own right, there is reason to believe that this simulation can run even faster. In an earlier investigation on their microVAX II, it was found that some of the table lookup functions performed during the simulation were taking an inordinate amount of time. The table lookup code had been written

for portability rather than performance, and what should have amounted to simple indirect memory references were transformed into several levels of nested subroutine calls with access functions and tables being passed down the call trace. By recoding these routines in VAX assembly, it was possible to realize an appreciable performance improvement - on the order of 30% for some of the subsystem modules. Similarly, the table lookup code in this simulation could be rewritten with moderate effort to provide greater performance.

More detailed analysis of the current simulation with the Gist performance analysis tool has helped to identify other areas in which some performance improvements could be made. For the performance analysis, a version of the simulation was augmented with a set of **event log points**. These points, which constitute timestamped user event definitions, are accumulated into an event log as a program executes, and can be reviewed later in any of a number of formats. Execution of programs with active event log points is typically within 20 to 30 % of the uninstrumented version, so that the general performance characteristics of the program is not appreciably altered.

Figure 7 shows one such review format. Four independent process traces, representing the four simulation processes, appear as a detailed state and event diagram. States are defined as the interval between two defined events, and can be defined to directly reflect the concept of simulation "tasks". In the plot shown, the "distance" on the time axis between successive events labelled "1" (in the event boxes), represent an instrumented simulation frame. The various stippled polygons represent state durations, the period of time between two appropriately defined endpoint events. Hence state **Rotor_1** is that period between events **Rotor_1Enter**, where the computations for module **Rotor_1** begin, and **S_Rotor_1Enter**, where the computation has completed and the state variables from the **Rotor_1** computation are written back ("scattered") to the shared state vector.

In this view, we can see that the **Rotor_1** computations, which had been spread across all four processors, comprise the largest component of simulation time. The state labeled **Barrier1**, which is evident in a number of places across all processors, represents the amount of time the respective processor spent waiting at one of the subframe barrier synchronization points. This is the amount of time spent waiting for one of the other processors to finish its subframe computation. The bottom trace in particular shows a substantial amount of time in this state. Referring back to the math model mapping diagram, we find that the fourth simulation process has nothing to do during the third simulation subframe. Simulation process 1 (the second trace up from the bottom) is summing the forces and moments, and integrating the equations of motion during this subframe, and is the gating computation.

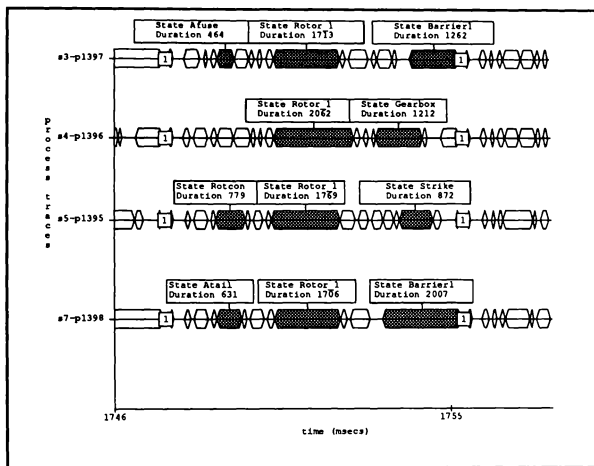


Figure 7. UH-60 simulation frame state diagram.

Gaps in the process traces are due to a number of factors. First, not all of the "states" the program goes through are being displayed in this plot. The user has the ability to indicate exactly which states and events to display at any one time. In this plot, we were interested mainly in the simulation subsystem components, and the respective gather (state vector read) and scatter (state vector write) operations. States representing executive interface logic, and various portions of the simulation driver logic are not displayed in the selected plot. Additionally, nX activities (which can be obtained and viewed along with user process information), and event logging overhead do not show up on the diagram.

Figure 8 provides an alternate view of process state durations, this time from a more global perspective. This plot shows the cumulative time spent in each of the various states over an entire event log. The cumulative time for all states in a given process will be somewhat less than 100 %, depending on the duration of the monitoring period relative to the actual execution time, nX activity, etc. It is the relative amounts of time a process spends in its various states that is generally of interest.

In this case, simulation process 4 spends nearly 45 % of its time waiting at barrier synchronization points. Simulation process 1 (the second from the bottom), on the other hand, spends only 10 % of its time waiting at barriers. Processes 2 and 3 fall somewhere in between. This disparity in the amount of time spent waiting at barriers is an indication that the processing loads are not

well balanced across processors during any one subframe. Process 4's lengthy barrier state duration can be attributed (to a large extent) to the fact that it has no computations to perform during subframe 3. But the significant barrier state durations for the other processes indicate that they too tend to spend periods of time waiting for one another throughout the course of the simulation frame. Process 1, for example, appears to complete its first subframe ROTCON computation slightly ahead of the SAS/FPS and ASFUSE/TROTOR computations performed by processes 2 and 3 respectively. Additionally, execution of its ROTOR_1 computation appears to proceed more rapidly than do the ROTOR_1 computations of the other processes, perhaps because the shared state vector is local to process 1. Process 2 and 3 both spend significant amounts of time waiting for process 1's force summation and equation of motion computations during subframe 3.

This end effect phenomenon indicates that the amount of work being performed across the processes in any one subframe is not even. Because of the relatively large task granularity in this version of the application, there is not a great deal that can be done to remedy this. While it is possible that portions of the GEARBOX and STRIKE routines could be parallelized, the additional performance obtained by such efforts is not required for real-time operation. However, in a more complex simulation, where there are more subsystems, and where subsystems are complex enough to be subdivided into parallel units, a better balanced distribution of work would be possible, and hence "utilization" would be improved. This we expect to

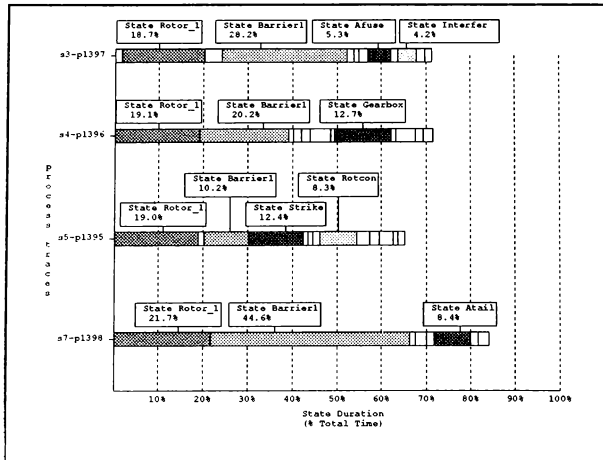


Figure 8. UH-60 process state durations.

see in the more complex aeroelastic blade model simulation.

Remember, however, that the focus in frame based simulations is NOT to optimize processor utilization, but rather to minimize (to some point at least) the simulation frame-time. Hence one is willing to live with less than optimal utilization in the interest of reducing the frame-time. For the rigid blade model simulation, the extra processing power appears difficult to tap.

Two of the areas which do look like candidates for

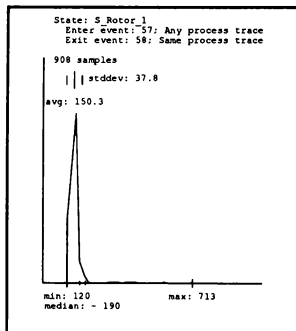


Figure 9. S_Rotor_1 state duration histogram.

improvement, however, include the barrier synchronization scheme and the state vector gather and scatter routines. Investigations into state histogram plots, such as that shown in figure 9, indicate that these areas might be improved by changing the underlying memory management model that was described above.

The barrier synchronization scheme employed in the simulation is implemented using a two-pass token based scheme. In the first pass, each participant looks in its own bin, waiting for its neighbour to deposit the token before passing it on. The first participant is special in that, upon entering the barrier, it immediately passes the token on. The second pass is used to "start" the processes back up using a similar scheme. The net result is that no process can get through the synchronization point until everyone has arrived, hence the term **barrier**. An optimal implementation of this barrier across multiple processors would have processes checking for the token in shared memory which is physically local, then depositing the token into another process' bin, which is physically remote. This reduces traffic to remote memories which might be involved in, e.g. the scattering of state variables at the end of a subframe, and thus improves the overall performance of the barrier. To use this trick, we need to provide a set of token "bins" that are logically shared, but physically distributed amongst the four participating simulation processors. This is a relatively straightforward task, and has been implemented on the TC2000 in other multiprocessing applications requiring barrier synchronization. Additionally, there are alternative synchronization algorithms that have also been

implemented, including one based on a distributed log combine operation that guarantees synchronization and startup in $O(\log n)$, rather than $O(n)$ time. For a four process barrier, the performance improvement afforded by the $O(\log n)$ barrier is small (20 ms vs. 30 ms). But for a larger number of processes (perhaps as few as 8), the difference is appreciable.

The state vector gather and scatter routines can also benefit from changes to the way in which the simulation's shared memory is managed. First, the overall memory bandwidth into and out of the state vector can be increased by interleaving it across the four participating processors. Memory interleaving is a technique that is used to increase effective memory bandwidth by providing parallel access to multiple physical memory modules through multiple access paths. In this application, we can provide for concurrent access by the simulation processes into disjoint regions of the state vector. This is particularly important at the beginning and end of subframe computations, where the processes are all attempting to read and write data in the shared state vector.

Second, the cross-switch **cache line fill** operation can be used to increase the bandwidth to a **shared cacheable state vector** in any one of the memory modules. A cache line fill operation reads or writes an entire cache line (16 bytes) in a single switch transaction, reducing transaction overhead and providing maximum memory bandwidth. In order to take advantage of this performance improvement, the shared state vector must reside in cacheable shared memory. Because scalable shared memory architectures such as the TC2000 rely on **software-based cache coherence techniques** (hardware coherence mechanisms become infeasible beyond about 16 processors, as the coherence bus saturates), this requires a coherence protocol to be established for access to the shared state vector. An analysis of the access patterns in the shared state vector and the implementation of a software coherence protocol for invalidating stale variables and flushing modified ones can be developed to raise the effective memory bandwidth into the shared state vector.

Subsequent Activity

As a follow on to the demonstration of the rigid blade UH-60 simulation at I/ITSC, the aeroelastic blade element model was ported to both the TC2000 and to a Silicon Graphics 4D/280 GTX, and demonstrated at the CSRDF⁴. The basic software architecture used on the TC2000 was retained, and most of the code was ported unchanged to the 4D/280 GTX. There were, however, some changes required in dealing with SGI's System V based shared memory interface. By appropriately encapsulating these O/S dependent characteristics into a more abstract interface, a portable shared memory model was developed and easily reimplemented on the TC2000.

Conclusions

In this paper, we have described a powerful real-time helicopter simulation development and analysis environment that has been developed and demonstrated on two distinct multiprocessor platforms. It incorporates engineering design and analysis tools and a multiprocessing software architecture that can be used concurrently in the evolutionary development and analysis of complex real-time simulations. Coupled with a pilot's station, it provides a unique setting in which simulation engineers, computer scientists, and pilots can work together in the design and validation of advanced real-time helicopter simulations.

We described the multiprocessor-based implementation of a blade element simulation of the UH-60 Blackhawk helicopter that demonstrates the ability to execute in real-time. Detailed analyses were performed to validate the functional characteristics and to investigate the performance of the simulation.

Finally, the prototype developed for the TC2000 was generalized to provide a portable multiprocess software architecture. This architecture centers on a machine independent process communication and synchronization model built on top of a small, well-defined machine dependent core. The feasibility of this approach was demonstrated in the subsequent implementation of an aeroelastic blade model at NASA Ames' CSRDF.

Acknowledgements

Parts of this work were supported by NASA grant NAS2-11960. The authors would also like to acknowledge the support provided by BBN Advanced Computers Inc.

References

- 1 Houch, J. A., Moore, F. L., Howlett J. J., Polecat, K.S., and Browne, M.M., "Rotor System Research Aircraft Simulation Mathematical Model," NASA TM-87629, 1977.
- 2 Gillman, H., "A Parallel Architecture for a Real-time Blade Element Rotorcraft Simulation," 1988 Aerospace Simulation Conference, The Society for Computer Simulation, ISBN 0-911801-28-6
- 3 DuVal, R. W., "A Real-time Blade Element Helicopter Simulation for Handling Qualities Analysis," Fifteenth European Rotorcraft Forum, 1989.
- 4 Hill, G., Du Val, R. W., Green, J.A., Huynh, L.C., "A Piloted Comparison of Elastic and Rigid Blade-Element Rotor Models Using Parallel Processing Technology," Sixteenth European Rotorcraft Forum, 1990

**SELECTING A REMOTE SENSING SUPPORT SYSTEM FOR TRAINING SIMULATION
AND MISSION REHEARSAL APPLICATIONS**

**Mark Bromley
Systems Applications Specialist
Planning Research Corporation
McLean, Virginia**

Abstract

The purpose of this paper is to present a methodology for selecting a remote sensing system for training and mission rehearsal applications. Although the methodology described here is for selecting a system with a specific application, the procedure could be applied to other applications as well.

The steps being taken by PRC to insure success in the selection of a remote sensing system include

- defining the project goals and objectives,
- presenting a proof-of-concept,
- defining products to be derived,
- identifying the software modules necessary to generate the products,
- describing the process flow with emphasis on defining parallel pathways,
- further decomposition of the software modules into specific functions necessary for product completion, and
- defining and weighting the discriminating factors that will be used to differentiate candidate systems.

completing development work on the prototype system.

The Rapidly Reconfigurable Data Base (RRDB) is a natural extension of the work performed to date on Project 2851. Its goal is to decrease the response time for generating simulator databases to meet a nominal 72 hour turnaround requirement for Special Operation Forces. Using commercial off-the-shelf (COTS) hardware and software technology in combination with existing geographic and intelligent database systems, the RRDB will

- extract elevation data from stereo imagery when necessary,
- apply realistic photo-texture to terrain and cultural features,
- extract pertinent feature attributes, and
- identify and/or develop automated and semi-automated techniques to accomplish the above. These may include the use of expert systems and artificial intelligence where applicable.

Multispectral imagery (MSI) has been identified through the Feature Extraction Study (FETS) completed by the USAF/ASD (Aeronautical Systems Division) and PRC as a promising data source to achieve these goals. With this in mind, the objectives of integrating MSI into the RRDB are

- to identify methods of quickly populating the Standard Simulator Data Base (SSDB) with feature attributes and elevation data generated by MSI,
- to provide a mechanism that will allow photo-texture to be used

Overview of Project 2851/RRDB Goals and Objectives

Project 2851 is a U.S. Air Force program (with tri-Service support) whose goal is to develop a database standard to support Department of Defense training simulators. A contractor team led by PRC is currently

- with polygon, gridded, and voxel based training simulators, and provide correlation among multiple imagery sets, map sources, and 3 dimensional models for natural vision, thermal infrared, night vision goggles, and radar sensor systems.

Development and Identification of MSI Products

MSI has the promising potential of rapidly populating the SSDB through wide area automated attribution. Such attributes as urban or built-up land, agriculture, range, forest, water, wetlands, barren land, tundra, and perennial snow and ice* can be derived readily from MSI data using standard statistical classification algorithms. Given more time and ground truth, additional attributes can also be identified but to a lesser degree of accuracy and with increasing levels of human interaction. One purpose of the MSI prototype is to help define what these additional features may be. Once extracted, these features can then be attributed and stored in the SSDB for later manipulations.

Other methods of feature extraction will also be investigated. These include semi-automated techniques that utilize the tasseled-cap transform, principal components, and line followers to help identify lineaments such as lines of communication (LOCs) and building edges.

Change detection algorithms will be utilized for updating existing databases in an RRDB environment. By focusing only on the changes since the last entry into the database, the operator will not have to re-extract all applicable features. Instead, the operator can concentrate on identifying and modifying only the new or changed features saving on production time and cost.

* USGS land use and land cover classification system for use with remote sensing data.

Manual feature extraction may take the form of delineating buildings, bridges, obstructions, target and surface material identification, and changes in LOCs.

General Electric RRDB Concept Demonstration and Final Report

On July 17, 1989 General Electric (GE) under contract to PRC concluded their proof of concept for utilizing MSI and other image source data to extract feature attributes and provide photo-texture information to an enhanced Compu-Scene IV image generator. The demonstration correlated DTED (digital terrain elevation data) and DFAD (digital feature analysis data) with MSI, constructed 3 dimensional building models, and applied photo-texture acquired from satellite imagery, airborne platforms, and ground photography to the models and gaming area. The GE demonstration provided the groundwork necessary to continue the project at the next level of detail.

Software Module Definition

Based on an analysis of the GE report, PRC has identified the image processing modules required in an RRDB environment. These modules are shown in Figure 1 of the appendix.

Process flow

Application tasking can be defined once the software modules necessary to production are identified. A breakdown of the tasks in a time sequence is derived using figure 1 as a guide. From this time sequence, both serial and parallel events are noted (see figure 2 of appendix).

The flow diagram is defined by levels to show both pipeline and parallel processing stages. Each level defines a common set of instructions to be performed by the software modules discussed in figure 1.

Parallel events may occur within a given level.

Level 0 denotes the starting point. This includes the formation of a database production plan, the acquisition of available imagery data, and a source assessment of the imagery.

Level 1 provides the elevation extraction process in lieu of existing DTED and will be stored in the SSDB for archival and future workorders.

Level 2 performs the raw image to image map conversion. This will provide correlation of the many data sets used in a given workorder.

Level 3 allows for quality control of the imagery prior to further processing. Upon completion of level 3, the data is in a format comparable or superior to available commercial products in the marketplace (i.e., most value added firms do not correct bad data for their geocoded products). The imagery will be stored in the SSDB as the nominal image set to provide a fallback position in case of catastrophic operator or system failure during production.

Once level 4 is engaged, the ability to accomplish parallel events takes form. Copies of the correlated image map are supplied to the feature extraction module as well as the photo-texture process. While one or more systems are performing the extraction and attribution process for features at coarse, medium, and fine levels of detail, a parallel system can be preparing the data as a true color drape to be used in level 5.

Level 5 will provide a visual mechanism to view and edit the imagery database in the SSDB. Three dimensional models, true color imagery, and elevation data need to be fused into a single composite image to verify feature and positional accuracy. This handshaking of visual inspection along with database generated assessments will further enforce the production of a quality product.

System Discriminating Factors for an Image Processor

Several key factors must be considered in the purchase of an image processing system for P2851/RRDB. The following factors will be used in the evaluation of COTS technology and are ranked in order of importance to the project.

1. Functionality. A complete image processing capability must be resident or have the ability to be integrated through the use of primitives or higher order functions.

2. Image Size. Image size processing limitations are a major concern to COTS hardware/software systems. Image sizes for P2851 will vary from under 512 x 512 pixels to over 32k x 32k pixels for radar and satellite image mosaics.

3. Execution time. Execution time is an important consideration for Project 2851. Factors to benchmark include storage, retrieval, and data transfers. Execution times are directly related to size of main memory, image memory, number of CPUs, availability of a co-processor (i.e., math or array processor), DMA (direct memory access), high speed/high bandwidth data bus, clock rate, and various hardware accelerators (i.e., hardware IHS transform boards).

4. User Interface. Editing functionality must be easy to perform and manage. Image data dropout, noise, bad scan lines, and bad header blocks are a fact of life in image processing. Correcting these problems should be as painless to the operator as possible.

All image processing functions should be useable on complete image sets, elements of an image set, or as blobs/masks occurring as subsets within an element. These blobs/masks can be designed as simple or complex geometric shapes. For example, a user may want to edit clouds on a TM scene over Lake Tahoe. The mask created will then be used to imbed non-clouded new imagery into the scene. This same mask can then be used to apply a custom lookup table (LUT) to

perform color balancing on the inserted imagery.

Some systems can, in a semi-automated way, locate bad scan lines and proceed to synthesize a replacement line by averaging the above and below lines. Others may apply a threshold filter to eliminate noise spikes. Features such as these are beneficial in a production work environment.

5. Numerical Computations. Math operations and how they execute are very important to data integrity. Ways in which a program executes and completes its math routines can make it perform faster by providing integer arithmetic or by using faster but less accurate approximations such as rounding of floating point values. Rounding after each math operation in a chain of operations can compound image data errors. Statistical classification algorithms depend on a high fidelity of radiometric values to extract features.

6. Utilities. Software utilities should include a capability to write custom code that can invoke each of the image processing functions exercised interactively. A batch processing capability is imperative to a production environment. Software to input non-standard sources of image data should be included.

7. Robustness. A major shortcoming of any off-the-shelf system is allowing the user to enter inputs that can crash the computer or terminate the program executing. All user inputs into the system should be anticipated in the software.

8. User Documentation. Well written, easy to understand cross-referenced user and system management documentation is essential. This includes a table of contents, index, system diagrams, flow charts, and meaningful error code diagnostics presented in a logical manner. Presentation of the document to novice users may provide the best test to measure user responsiveness.

9. On-line Assistance. A good system will provide on-line help documentation to allow the user to answer most questions without

searching through written documentation. These functions can and should refer to specific manuals for further clarification and provide concise written examples showing how a process is executed.

10. Sample Database. A good system will provide a sample image database for novice users to immediately familiarize themselves with the new system. Examples may include a sample 512 window that is used repetitively as a working tool when demonstrating various program functions.

Functional Discriminating Factors for an Image Processor

Image processing software functions can be analyzed in two ways. There are program modules which can be looked at in simple black and white terms, and there are other routines that allow for considerations to be given to unique vendor implementations of these basic tools.

Functionalities such as math programs that do +, -, /, * for both point data and matrix data, logical operations such as AND, OR, XOR, NOT, and input/output routines are fairly straightforward. Either they work or they don't work. Other functions allow for additional weighting factors to enter the analysis. Not only must these functions perform correctly but they can add much desirability to the workstation by the way they interact and display pertinent information to the operator. For this reason these functions have been separated out from the rest of the image processing modules.

Included is a list of modules for which additional requirements or considerations will be defined. Since these functions can be implemented in several ways, they will be judged by their user friendliness and system implementation. Assigning high, medium, and low weighting factors in a matrix format will aid in gauging user concerns when comparing various remote sensing systems.

- Procedure Statistical Feature Extraction
- Procedure Manual Feature Extraction
- Procedure Map Projections
- Procedure Rubbersheet and/or Geoposition
- Procedure IHS (intensity, hue, and saturation) Transform
- Procedure Histogram
- Procedure Color Balancing
- Procedure LUT Management
- Procedure Mosaic
- Procedure Imbed (cookie cutting/image splicing)
- Procedure Data Repair
- Procedure Principal Components Analysis
- Procedure FFT (Fast Fourier Transform)
- Procedure Filter
- Procedure Image Synthesis (for algorithm verification)

Acceptance Testing

Testing of image processing software does not have to be difficult although it should be exhaustive. Below is a sample test of logical operations for a remote sensing system. Notice that the final result should be a blank screen. If the screen is not blank (test by contrast stretching the results or run a histogram of the image), then the system may not be functioning correctly. Such tests can be devised for many of the image processing functions.

Sample test for Logical Operators (graphics only). Test for screen to screen, screen to disk, and disk to disk operations.

- 1) Interactively mask water from imagery (create WATER mask); leave some noise (shoreline effect).
- 2) Edit noise from WATER mask while blinking graphics in zoom and pan mode (provides a hardware and software test).
- 3) Reload original WATER mask.
- 4) Interactively mask land from same imagery (create LAND mask).

5) Create NOT (LAND).

6) Perform:

- a. LAND AND (NOT LAND)
= blank screen
- b. (WATER mask) AND (LAND mask) = noise pixels
- c. (WATER mask) OR (LAND mask)
= full image scene
- d. NOT((WATER mask) XOR (LAND mask)) = noise pixels
- e. subtract results of b from d
= blank screen

Conclusion

Understanding the problem, deciding on the output products, and defining the hardware and software functions necessary to derive these products is the key to successfully evaluating and selecting a remote sensing system. PRC intends to apply a systematic approach to insure objectivity and cost effectiveness in our selection of image processing technologies for the P2851/RRDB production system. Furthermore, PRC would welcome all vendor and end-user inputs into the selection criteria.

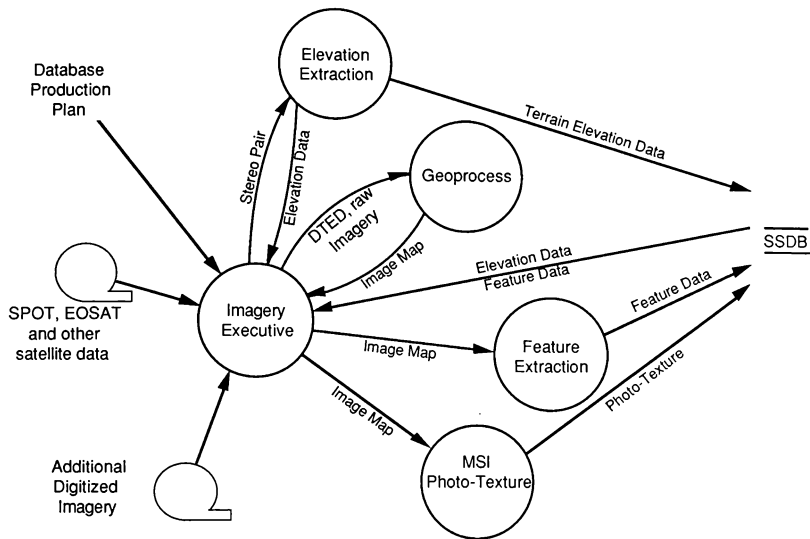


Figure 1

Software Module Identification

Level	Task	Parallel Tasks
0	Define gaming area Acquire raw data, source assessment	
1	Stereo elevation extraction Accuracy evaluation SSDB input (DTED/DEM)	
2	Precision geocode and orthorectify (geoprocess) - DTED, raw imagery inputs - sensor modeling with ground control Accuracy evaluation	
3	Bad data edit SSDB input (orthorectified image map)	
4	<div> <p>Feature extraction</p> <pre> graph TD FE[Feature extraction] --> SC[Statistical Change Detection] FE --> M[Manual] FE --> SA[Semi-automated Tasseled-Cap] FE --> LF[Line followers] SC --> AE[Accuracy evaluation] M --> AE SA --> AE LF --> AE AE --> SSDB[SSDB input (feature attributes)] </pre> <p>Statistical Change Detection</p> <p>Manual</p> <p>Semi-automated Tasseled-Cap</p> <p>Line followers</p> <p>Accuracy evaluation</p> <p>SSDB input (feature attributes)</p> </div>	<p>MSI Photo-texture</p> <p>Histogram equalization (color balancing)</p> <p>Create gaming area</p> <ul style="list-style-type: none"> - mosaic imagery - cut out area of interest <p>Cloud replacement</p> <ul style="list-style-type: none"> - create cloud mask - replace clouds with new data - color balance seam <p>Adjust solar illumination</p> <p>Contrast stretch imagery</p> <p>Edge enhance image set</p> <p>IHS transform, merge</p> <p>Inverse IHS transform back to RGB</p> <p>SSDB input (true color)</p>
5	<p>Previewer stage (verification of features and models)</p> <ul style="list-style-type: none"> - tie image to SSDB elevation data - imbed 3D feature models - overlay SSDB 2D culture features for correlation - synthesize still scene perspective view 	

Figure 2

Process flow diagram

PROJECT 2851 DATA BASE STRATEGIES
FOR SIMULATOR CORRELATION

Ken Oda
Project Manager
Planning Research Corporation
McLean, VA

Abstract

Within the DOD simulator community, there is a growing emphasis on improving correlation among sensor displays within a weapons system trainer, and also among trainers networked to simulate force-on-force engagements. Given technological and fiscal constraints, absolute correlation among all simulators is an unrealistic goal. What is required is a well-defined process for prioritizing correlation requirements and making technical and cost trade-offs affecting the degree of correlation achieved. The standard data bases to be provided by the Project 2851 system represent a significant step forward towards achieving this objective.

Background

Project 2851 (P2851) is an Air Force R&D program tasked with developing standardized data bases and transformation software in support of DOD training simulators. The primary goal of P2851 is to reduce the costs of duplicative data base development across DOD training simulator programs. But another major benefit of increased standardization will be better correlation among sensor simulators, both within a single weapons system trainer and across multiple trainers networked to simulate engaged forces.

The typical trainer today consists of independent computer image generators (CIGs) used to simulate each of the major sensor

systems, usually including a visual (out-the-window) and a navigational radar simulator, and sometimes also infrared and other sensors. Each of the CIGs typically has its own unique digital data base representation of the area of the earth (gaming area) over which operations are to be simulated.

For technical reasons having to do with vendor-specific strategies for optimizing real-time performance of the CIGs, the contents of simulator data bases may vary considerably, even though the same gaming area is being modeled, and the same source materials are used. These differences include variations in terrain representation, feature content, data resolution, and levels of detail.

For example, some simulators internally represent the earth's terrain surface as a grid of elevation values, while others represent it as a mesh of polygons in 3-D space. In the case of the gridded terrain, different simulators may be able to handle different resolutions (spacing) of elevation values. In the case of the polygonized terrain, some simulators may be able to handle rectangles, other only triangles; some simulators may prefer regularly spaced triangles, while others can handle irregularly spaced and shaped triangles; and each simulator has its own maximum number of terrain polygons or edges which it can continuously render in real-time.

In the area of feature content, major variations are possible due to

the varying techniques used to meet real-time polygon budget constraints. Different vendors will "throw away" different features as being unimportant, reduce the level of detail of the features they keep, use vendor-specific generic versus real-world models, and apply different forms of texture mapping.

These unique data base strategies are the natural result of an innovative and competitive industry working to overcome limitations both in real-time image rendering technology and in their customers' checkbooks. Within the simulator industry, it is a given that the customer's desires for data base fidelity and detail will always exceed what is technologically possible for the price the customer is willing to pay. Therefore, each vendor seeks market share through relative price/performance advantages gained via leapfrog advances in hardware, software, and data base technologies.

Due to differences in operating characteristics, radar simulator vendors have historically delivered far more detailed data bases than their visual system counterparts. This is possible because real-beam radar displays have a built-in sweep interval, greatly increasing the number of rendering computations which are possible relative to a real-time visual display. As a result, every training simulator integrator faces a correlation problem between his radar and visual displays. In most cases, correlation is achieved only by sacrificing some of the potential capability of the radar system.

Another aspect of the correlation problem is that, given present technologies, the creation of simulator data bases is far from fully automated. Each data base is created interactively, by specialists called data base designers or data base modelers, which by definition means there is a subjective element to data base content. The subjective

element is further accentuated by the fact that the services understandably involve pilots and crew members in the evaluation of data bases, and each evaluator brings his/her own perception of what is important to the mix. Different evaluators may require individualized and thus inconsistent changes to data base contents, ranging from issues of feature significance to the fine-tuning of colors in the scene.

Finally, it must be recognized that simulator data bases are different because their intended applications are different. A data base being built for tank simulations will tend to emphasize different aspects of the gaming area than a data base supporting fighter aircraft simulations over the same area. Yet, the day is coming when both systems will need to be integrated, and hence correlated, in a networked simulation.

Theoretically, one could require that all simulators use an identical, universal data base for any given gaming area, containing all data which may be of use to any simulation. But this would be impractical, guaranteeing that the simulators would be overwhelmed by the sheer volume of data. This will be particularly true with the growing emphasis on the use of low-cost image generators for affordable simulator networks. As a practical matter, an approach is needed that accepts the existence of simulators with a wide range of data base capacities.

Requirements for Correlation

If we accept, for the foreseeable future, that forcing all simulators to use identical data bases is an impractical solution, then it follows that the requirement for correlation becomes a matter of degree, one of the many trade-offs that must be made each time a training system is configured.

The key implication is that every simulator application must define for itself what kind of correlation is more or less important for that application. For example, precise correlation of terrain is critical for helicopter or tank combat simulations where the occulting effects of terrain are fundamental to the action being simulated. High-speed aircraft also depend to some extent on the occulting effects of terrain, but the requirements for precise correlation would not be as high.

Feature correlation would be more or less important depending on a variety of factors. For example, correlation of features at a navigation point or in a target area would be far more important than in an area of general ground clutter. Features that would attract the eye, such as lights or bright radar reflectors, should be correlated before relatively undifferentiated features. Different features would assume greater significance depending on the environment being simulated, such as night versus daylight operation. Under any conditions, hazards and obstructions need to be correlated for all the players in a networked simulation.

Since there are many possible combinations of simulators and gaming scenarios, this implies that a highly flexible and responsive approach to data base generation is required. The data base methodology must encompass some means for expressing relative priorities of data base contents, as well as a systematic approach for making trade-offs based on those priorities.

The key trade-off made possible by a prioritized approach would be that between correlation and CIG performance. As previously mentioned, it has sometimes been necessary to sacrifice some of the potential detail in a radar simulation to ensure correlation with a visual simulator. In the general case, when correlating any

combination of CIGs of varying capacities, it is not possible to optimize every CIG's performance concurrently. The goal of the trade-off would be to achieve the optimal balance of correlation and CIG performance.

Taking a broader view, the requirement for data base correlation can be seen to extend beyond the confines of pure training simulations. Particularly as simulators begin to be used for mission rehearsal, there are obvious advantages to having correlation among data bases used for mission planning, previewing, and rehearsal systems. Consistency of data bases will permit a series of useful feedback loops. The mission planner will be able to preview his plan, and the crews will be able to rehearse the plan, with each step either validating the plan or identifying areas for improvement. Such feedback would be pointless if the data bases are not correlated in key respects.

Keep in mind that, even in this context, a requirement for correlated data bases does not imply a need for identical data bases. Some of the data needed by a planner are not needed for crew training, and vice versa.

But more fundamentally, it is important not to inhibit technological advances and economic competition by having an overly rigid concept of data base standardization. A standard data base should be flexible enough to be a boon to innovation and competition, rather than a constraint.

P2851 Data Base Strategy

The P2851 strategy for supporting better correlation among simulators is based on flexibility and prioritization of data base contents. In this context, the key features of the P2851 prototype system architecture are described in the following paragraphs.

To begin with, there is a Standard Simulator Data Base (SSDB), which is intended as a repository of all potentially needed terrain, culture, model, and photo texture data. The SSDB will be built from standard Defense Mapping Agency (DMA) products, but will be enhanced to meet the specific needs of the training simulation community, using a variety of sources such as drawings, charts, photographs, and ultimately digital imagery. The maintenance of an SSDB supports correlation by providing a common starting point for all users of P2851 data bases. Within the SSDB, data will be maintained at several correlated levels of detail.

As discussed earlier, a data base that contains information needed for any possible simulation will overwhelm any given simulator. Thus, the P2851 architecture recognizes that the SSDB must be significantly filtered and transformed for use in a specific simulation.

This is where the Common Data Base Transformation Program (CDBTP) comes into play. The CDBTP is responsible for extracting data needed for a particular simulator application from the SSDB and transforming it into a data base product called a Generic Transformed Data Base (GTDB). The contents of a GTDB may be tailored via the specification of a wide variety of transformation parameters. Among them are a few key CDBTP parameters which are expressly intended to enhance data base correlation.

For terrain, the user may request either gridded terrain, polygonized terrain, or both, in a GTDB. It is often the case that a radar system will process gridded terrain, while a visual system requires polygonized terrain. When correlation between the two is desired, the user may specify that the elevation values in a GTDB terrain grid be derived from GTDB polygonized terrain rather than from the SSDB source grid.

Correlation between two GTDBs with differing numbers of terrain polygons is also enhanced by the CDBTP's use of a polygonization algorithm that incrementally adds vertices until either a goodness-of-fit parameter or a maximum-number-of-polygons parameter has been met. The incremental nature of the process means differently specified GTDBs will still tend to have the most important terrain vertices in common.

For culture, the user may specify lists of feature categories called a Keep List and a Delete List, which the CDBTP will use to prioritize which features it will retain or throw away in the process of filtering culture to meet a polygon budget. Thus, even if two GTDBs have differing parameters for maximum number of culture polygons, they will tend to share those features identified as most important for correlation.

In addition, the user may specify areas of relatively greater feature density, called Islands, within the overall gaming area. Thus, two different GTDBs may have different culture polygon counts, but would tend to have relatively greater or lesser detail in common areas. Islands may be used to concentrate detail in high-interest areas such as navigation points or target areas.

From an architectural standpoint, use of P2851 data bases improves prospects for correlation simply by the fact that they are centrally produced data bases, eliminating some of the subjective variations possible when data bases are built independently by several data base modelers. (In truth, since the SSDB is maintained interactively, there is still room for subjectivity, but it will be a consistent subjectivity.)

On the other hand, the P2851 system concept calls for the GTDBs to go through a final formatting step, which converts the data bases into simulator on-line formats. This

approach recognizes that on-line formats are CIG-specific and usually proprietary. It also has the benefit of permitting last-minute updates to the data base. But any unique changes will defeat the effort to achieve correlation. This implies that changes to GTDB contents ought to be minimized, and, when they are necessary, ought to be applied to all the data bases being correlated.

As a final caveat, it must be noted that data base content is by no means the only significant factor affecting CIG correlation. CIGs also vary in their ability to model what are generally referred to as special effects, including independent moving models, threats, weapon impacts, collision impacts, and environmental variations. Thus, even given consistent data bases, two CIGs may behave differently under certain circumstances. P2851 data bases obviously will not change this situation.

Conclusions

Given technological and fiscal constraints, as well as the need to foster innovation, it is not desirable to require absolute correlation of simulator data bases. However, a requirement for ever increasing degrees of correlation has emerged in parallel with interest in mission rehearsal and networked simulators.

The keys to achieving a desired degree of correlation are (1) a definition of what kinds of correlation are important to a simulator application; and (2) consistent application of correlation priorities in selecting data base contents.

Project 2851 supports simulator data base correlation by providing: (1) a common starting point of correlated source data; (2) a flexible set of transformation parameters for specifying correlation priorities; and (3) a common point

for decision-making regarding data base contents.

P2851 data bases cannot guarantee correlation among simulators, but they will offer significant opportunities for improvement over continued use of independently created data bases. As an emerging standard, P2851 seeks constructive feedback. Readers are invited to submit ideas to the author for improving P2851 support for data base correlation.

Acknowledgment

Project 2851 is sponsored by the Aeronautical Systems Division of the Air Force Systems Command (AFSC/ASD) located at Wright-Patterson AFB, OH.

A RAPID DATABASE CONFIGURATION SYSTEM USING MULTISPECTRAL IMAGERY

Jeffrey A Lickenbrock
Senior Engineer
McDonnell Aircraft Company
A Division of McDonnell Douglas Corp.
Dept. 252, M/C 0641461
P.O. Box 516
St. Louis, MO 63166
Phone: 314-234-0303
FAX: 314-232-7972

Karl A Spuhl
Senior Principal Technical Specialist
McDonnell Aircraft Company
Dept. 252, M/C 0641481
Phone: 314-233-8979

R. E. Brown
Staff Manager - Senior MDC Fellow
McDonnell Aircraft Company
Dept. 280, M/C 0641481
Phone: 314-233-6813

ABSTRACT

McDonnell Aircraft Flight Simulation has developed a system for creating high-detail real-world simulator databases from satellite imagery. This system is operator interactive and is usable in both a stand-alone mode and as an enhancement tool for an autonomous feature extraction system. A sample database tile is presented as well as a simulated radar image generated from it.

operator interactive photobased feature extraction system based on an image processor. We are currently using a Vicom IP8500 image processor. By configuring our system to use the array processing capabilities of an image processor, we have been able to achieve very high speed feature extraction.

DATA SOURCE

Our system uses multispectral satellite imagery as its data source. Multispectral satellite imagery was chosen because of its high level of detail and its ready availability over the globe. Table I lists the major specifications of the two most common sources of civilian satellite imagery. The imagery used in this report was created using data from the French SPOT satellite. This data was obtained in the form of Computer Compatible Tapes (CCT) containing the digitized spectral bands IR, Red, and Green as well as high-resolution Panchromatic.

INTRODUCTION

In recent years, technological advances have greatly increased the capability of flight simulation systems. These advances have made possible a greater level of detail or "realism" of the overall simulation. As the level of realism improves, increased customer emphasis is placed on the quality of the database through which the simulator flies. Whereas once a mechanically scanned terrain board was a sufficient database, systems of today require geographically large, highly detailed real-world databases. These databases must be generated quickly, provide correlation between visual and sensor displays, and be current. The most widely used data source, Defense Mapping Agency (DMA) Digital Feature Analysis Data (DFAD), is produced in several levels of detail over various parts of the world. DMA, however, does not currently have a requirement for a high detail database covering the entire world. Therefore, a method of producing high detail databases is needed which is both rapid and which is based on a current and readily available data source.

TABLE I: Satellite imagery specifications.

<u>SATELLITE</u>	<u># BANDS</u>	<u>RESOLUTION</u>	<u>CCT FORM</u>
SPOT	3 (XS)	20 m/pix	Yes
	1 (Pan)	10 m/pix	Yes
LANDSAT	7 (TM)	28.5 m/pix	Yes

Notes: XS - Multispectral
Pan - Panchromatic
TM - Thematic Mapper

APPROACH

McDonnell Aircraft Flight Simulation has developed a database generation system which addresses this problem. We have developed an

FEATURE EXTRACTION ALGORITHM

The algorithm on which our system is based relies on the fact that different types of materials will respond with distinct intensities throughout the various spectral bands.

Therefore, by filtering a certain intensity range from a given spectral band, we can isolate all features that have the same response in that portion of the spectrum. While filtering from a single band will roughly isolate a given type of feature, the selectivity of the process becomes much greater with the addition of more spectral bands. Our system was initially designed to handle 4 spectral bands, but is readily expandable to 7 or more bands.

A block diagram of our system's filtering algorithm is shown in Figure 1. It shows as its inputs, the multispectral and panchromatic bands of a SPOT satellite image. Because of the resolution difference between the SPOT multispectral and panchromatic image pixels, when both are used together, the multi-spectral image bands are zoomed using standard pixel replication techniques and registered to the panchromatic band prior to input to the system.

The first operation performed by the system is a variable sized filtering area-of-interest designation. When the images are loaded, they are displayed on a monitor in a false-color representation (Figure 2a). Using a trackball connected to the IP8500, the operator outlines some variable size and shape area-of-interest on the displayed image (Figure 2b). This designates the area over which the filtering rules will apply. Through this selective area filtering technique, unwanted artifacts, which would pass through a filter operating over the entire image, are ignored. This allows wider filtering limits, and thus more accurate feature extraction.

Once this filtering area-of-interest has been designated, the screen zooms into the selected area and the operator is given the option of performing a histogram equalization. The histogram equalization sharpens the contrast of

the displayed image so as to aid the operator in visually distinguishing features. This operation affects only the copy of the image stored in display memory, not the images stored in working memory used for the filtering operations.

Again using the trackball, the operator then positions a cursor over the feature which is to be extracted (Figure 2c). The pixel values at the cursor location initialize the band filter values. These filters are then instantaneously applied to the image bands. Their outputs are combined using a series of logical operators (L1 to L3 in Figure 1) to form a single filtered output. For added selectivity, these operators can be individually chosen from any of several logical operators (AND, OR, NOR, NAND, XOR). The logically combined output is then instantaneously displayed as an overlay on the original image (Figure 2d).

Next, using the keyboard arrow keys, the operator can vary the limits on each band filter and/or the logical operators while instantaneously viewing the new logically combined output. This instantaneous visual feedback affords great selectivity in the filtering operation.

When the operator decides that the extracted feature has a level of detail appropriate for the database being created, the feature is assigned a DFAD-compatible Feature Identification Code (FIC), Surface Material Code (SMC) and feature height and inserted into the output database tile (Figure 2e). Upon completion (Figure 2f), the feature-extracted database tile is stored to disk in a gridded pixel format. Because the database tile is stored in this gridded format and the geographic size of each pixel is determined by the input data source, no feature length and width need be input at this stage.

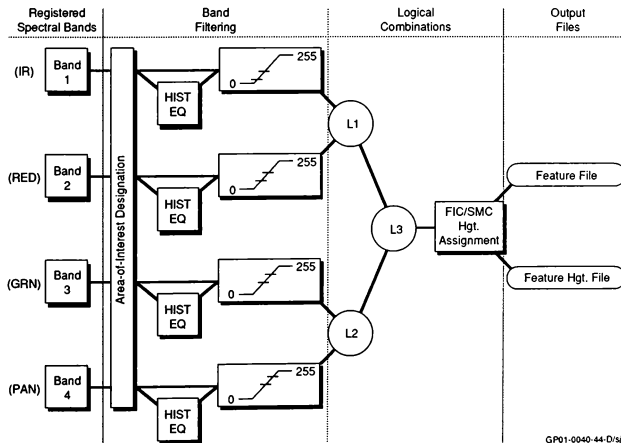


Figure 1. By Choosing Appropriate Filter Bands and Logical Combinations, Features Are Instantaneously Selected

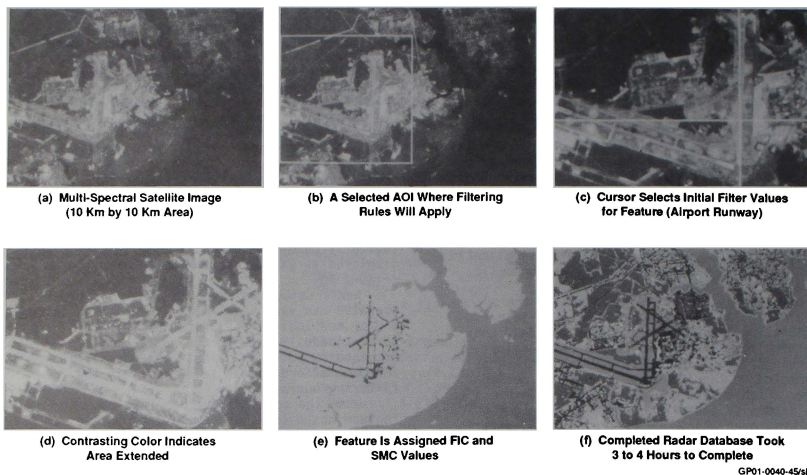


Figure 2. Feature Extraction System Provides Instantaneous Visual Feedback to Operator

A tile of the complexity of that shown in Figure 2(f) can be generated using this method in approximately 3-4 hours. As this is a 512 x 512 pixel tile with a resolution of 20 m/pixel (using 20m/pixel input data), this equates to 30.53 sq. nm / 4 hr or 7.63 sq nm / hr. This can be compared to the DMA standard of 8 sq. nm / 125 hrs for DFAD Level II. A less detailed tile can, of course, be created in much less time.

high degree of spatial accuracy. Step 3 then adds additional DFAD variables to the file. Finally, step 4 adds the appropriate DFAD header blocks to form the DFAD format tape. At this point, the features from the gridded database are represented as DFAD areal features. This conversion provides a common format which most visual and sensor image generators can take as input.

FORMAT CONVERSION

The gridded database output from the feature extraction system can be used as is, or converted to a DMA DFAD tape format. This conversion takes the form of a four-step process as shown in Figure 3. The first step converts the gridded database features into a polygon vertex list file. The second step then reduces the number of vertices in large polygons while retaining a

A PROTOTYPE AUTONOMOUS SYSTEM

Up to this point, our system has been presented in a stand-alone configuration in which its input is the original spectral image. Another approach would be to use this system as an enhancement tool for the output of an autonomous feature extraction system. This concept is illustrated in Figure 4.

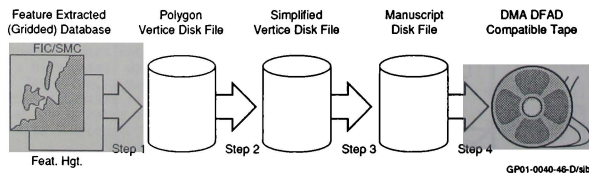


Figure 3. Feature Extraction (Gridded) Database Tiles Are Converted to a DMA DFAD Compatible Format

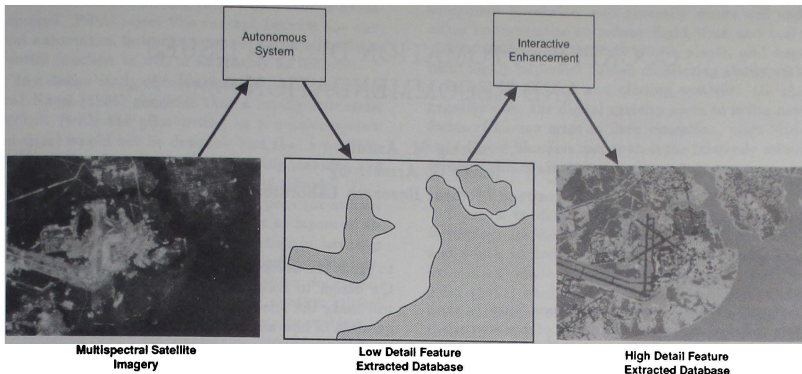
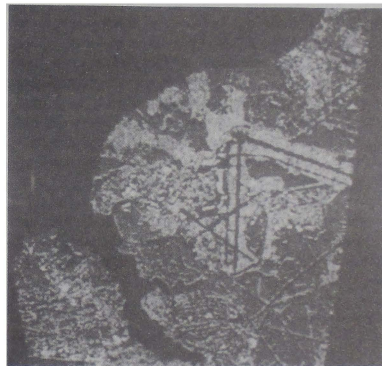


Figure 4. Interactive System Used as an Enhancement Tool for an Autonomous System

GP01-0040-47-5/b

A system configured in this way would perform rough, but high speed, feature extraction over large areas of the database using the autonomous section. At this point, the feature extracted database would be suitable for use in areas outside of the high level-of-detail corridors of a radar database. These high level-of-detail areas would then be formed by passing sections of this rough database through the interactive feature enhancement section. This section would operate as described earlier, but rather than creating the entire database tile, it would simply be adding detail to those tiles already created.

The autonomous section could consist of any of a number of emerging technologies including neural networks, special purpose expert systems, or standard programming languages such as C or FORTRAN. We decided that since there is currently a large amount of research progressing in these areas, it would be best to adopt a "wait and see" attitude to determine which technology seems the most promising for incorporation into the front end of this system.



GP01-0040-48-5/b

Figure 5. Realistic SAR Image From Database of Figure 2 (f)

CONCLUSION

This paper described two approaches to the feature extraction problem, both utilizing a common software tool developed by McDonnell Aircraft Company. This tool has proven very useful for our in-house needs. While its output is currently too fragmented to directly form a visual database, it builds excellent radar databases as evidenced by the simulated SAR display of Figure 5. This image was created using the database of Figure 2(f). It has also been used to develop databases for static map insets and, with sufficient processing, a correlated visual database should be able to be constructed from it.

COCKPIT AUTOMATION DESIGN ISSUES AND RECOMMENDATION

Craig M. Arndt
Harry G. Armstrong
Aerospace Medical Research Laboratory

Abstract

Automation has become a significant issue in the design of fighter cockpits. The design of modern cockpits requires the use of advanced digital avionics and automated systems to perform the electronic warfare and flight control tasks in high performance aircraft, as well as a host of other tasks dictated by the demands of threat enhancements and use in multiple roles. The evolution of the digital avionics permits enhanced capabilities, but with the consequence of added complexity. This paper investigates the consequences of implementing cockpit automation into new and existing cockpit designs and the operational considerations. Three phases of automation implementation are defined, as a function of how tasks are delegated and during what phase of the design and operation of the aircraft the delegation takes place. These phases are 1) during the Design phase, 2) during the Configuration phase, and 3) during the Operational phase of the aircraft subsystem definition. The paper also describes a systems approach to automation implementation as a function of design application. 1) Total integrated cockpits, 2) partly integrated cockpits, and 3) Insertion into already existing cockpits (retrofit efforts)

1 INTRODUCTION

In this paper I will not attempt to solve the problem of automation implementation. Instead the goal of this paper is to highlight some of the key issues in automation implementation and to present a new prospective and method for addressing practical automation implementation projects. The literature on automation in workstations suggests that automation of functions cannot be treated as an all or none concept. The idea of levels of authority granted to the automated system has become widely accepted. The question now becomes not only how much authority

to grant the automated system, but at what point in the design to make the allocation decision and who will make the allocation decision. Then there is the question of how automation decisions will impact the total system integration and performance (including the human operator). I will address some of what I feel are the important issues in cockpit automation which need more work and maybe a new approach.

In the methodologies section of the paper three levels of automation implementation are defined, as a function of how tasks are delegated and during which phase of the design and operation of the aircraft the delegation takes place. These levels are 1) during the design phase, 2) during the configuration phase, and 3) during the operational phase of the aircraft subsystem definition. The paper also describes a systems approach to the automation implementation insertion as a function of the design application in which the automation effort will take place: 1) Totally integrated cockpits/projects, 2) Partly integrated cockpits/projects, and 3) Insertion into already existing cockpits (retrofit efforts).

2 PROBLEM

The problem is how to go about designing overall systems, such as an aircraft with automated sub-systems as a useful part of the system, without degrading the performance of other critical system components (eg. the pilot). The current design philosophy that implements automation based on available technology or apparent performance increases, over the human operation, has met increasing pilot acceptance problems.

The designer's solution to this situation has been the promotion of a concept which redefines the role of the pilot as an aircraft system manager rather than as an aircraft controller. Carried to its natural conclusion, this concept would automate the flight path control of the aircraft in order to free the pilot to per-

form other mission critical tasks which cannot be automated. Pilots reject this concept because the current automation technology cannot perform the flight control function as well or as reliably as the pilot.

In a design study of commercial aircraft, Chambers and Nagel (1985) conclude that a totally automatic cockpit (with the pilot acting as a passive system monitor) would not be desirable and that a partially automated cockpit (with the pilot doing planning and and procedural tasks) would be more reliable. They claim that "humans make relatively poor (or at least unreliable) passive monitors, subject to lapses of attention and sensitivity". This solution is basically contradictory to an overall systems design procedure. By optimizing the automation sub-system, we are adversely effecting the performance of another critical component of the system. Another major problem in the insertion of automated systems into old or already existing designs. Many application areas, including aircraft design, are looking at 1) upgrading capabilities, and/or 2) standardizing with newer designs, by adding automated systems to their older equipment. This can not only cause integration problems, but the requirement to completely re-evaluate the operator interfaces, training, and procedures. Lastly these more automated designs raise considerable concerns in the area of fault detection and maintainability. If the system operator is simply monitoring the performance of the aircraft he quickly loses his ability to follow the status of particular systems sensors and components and his ability to be an effective system trouble shooter. This can only have a negative result on maintainability and system confidence.

3 BACKGROUND

The literature on automation suggests that different guidelines, or design rules, may be applicable to different levels of automation or function criticality.

Cockpit automation is a subject that evokes considerable controversy among users, manufacturers, and regulatory agencies. Many see cockpit automation as a great boon to safety, removing human error at its source and replacing fallible humans with virtually unerring machines. The critics view automation as a threat to safety, replacing intelligent humans with devices that are both dumb and dutiful, a dangerous combination.

There appears to be ample evidence to support both positions; however as usual, the truth is undoubtedly neither but some combination which varies based on the application and the skill of the designers. On the positive side, the new digitally based

equipment is reliable, and generally works well and offers opportunities to reduce flight time and costs (fuel management), operate power plants, and augment highly imperfect human monitoring ability with a variety of warning and alerting systems. On the negative side, the digital systems seem to invite new forms of human error in their operation, often leading to gross blunders rather than the relatively minor errors which characterize traditional systems.

There is to date no systematic, widely accepted technology for determining which tasks to automate and which to assign to pilots. Numerous attempts have been made to define a systematic method. Fitts (1951) qualitatively characterized those functions performed better by humans than machines and those functions performed better by machines than humans. Since Fitts study several more detailed procedural guides, computer simulations, and information support systems have been developed for use in the allocation of functions in systems development. Each of these has features that can be applied usefully to systems development but, for various reasons, none of these has become a widely used tool of system design (Price, 1985). Probably the most important aspect of function allocation is summarized by Price (1985): Human and machine performance are not always antithetical. It cannot be assumed that if a human is a poor controller, then a machine must be a good one. There are functions that neither man nor machine do well, and there are functions that either man or machine can do within the requirements of the system.

Traditional design philosophy says we should automate functions which the system can do better and faster than the operator. On the other hand, we should leave to the pilot those things which he/she can do best. The pilot excels in assessment of complex situations, decision making, intuitive judgment, and complex pattern recognition (Eggleston, 1987).

In the future, integrated digital avionics will have the capacity to produce more information than the pilot needs or can use. To develop a useable system, even minor functions may have to be automated to keep pilot workload at a manageable level, so that the pilot can maintain attention on the critical aircraft and mission functions. Traditional design philosophy will, therefore, have to be replaced by one which emphasizes automating functions on the basis of their effect on system capability, reliability, and effectiveness.

4 AUTOMATION ISSUES AND APPROACHES

In the rush to create deterministic rules and procedures to make task allocation decisions and design decisions, a number of important issues are not being addressed in the efforts to implement new automation systems. It is important to remember that we are by and large dealing with very complicated systems and that simple rules and procedures may not always be adequate for describing the systems interactions and performance. As a consequence of addressing automation at a high level, many of the outstanding issues deal with systems interactions and operational aspects of automation.

4.1 Systems Considerations

The problem of pilot workload has been of considerable interest. Workload problems are generally discovered in one of two ways, either when the workload of a particular situation is so great that the pilot cannot perform critical tasks, or when boredom creates a lack of attention or vigilance during which critical tasks are not performed. The present methods for solving workload problems have had mixed results. These methods typically include: training, changing procedures, selection of pilots, changing the composition of crews, and improvements to the human engineering in cockpit design (Air force study, 1982). What is needed to make a significant improvement in solving and preventing workload problems is not just a reduction in overall workload, but a change in the way we construct systems under workload constraints. Ideally the workload level on a human operator, for optimum long term performance would be at some manageable level of difficulty and changing somewhat with time. We should make an effort to achieve these workload conditions whenever possible. Automated systems need to be designed to vary pilot workload in order to maintain attention, and level of pilot skill, as well as enhancing the understanding of current aircraft situation and what the automated systems are doing.

The concept of mutual awareness is critical to any complex partnership whether it be a marriage or a partnership between an intelligent piece of equipment and its operator. The operation of an aircraft is understood to be a partnership between the aircrew and the aircraft (automated systems), therefore it is vital that both be aware of the others intentions, goals, and current state of operation. Changes or additions to training systems have often been the proposed solution to a variety of problems that have developed with

the operation of automated systems. It is very important therefore that systems be set up such that the operator can understand and predict the performance of the automated system. Unpredictability is the direct opposite of safety in an automated system. As National Transportation Safety Board reports have reported (ref. 5) in a number of accidents, any time the aircraft automation performs in a manner which the aircrew does not expect, disasters can happen.

4.2 Automation Residual Tasks

In the 1982 Air Force Studies Board Report on Automation in Combat Aircraft the author states (ref. 1) " If the task falls within the performance domains of both man and machine, the designer can safely opt for either.". This does not take into account "Automation residual tasks" or "complementary Automation tasks". In most cases a system cannot be total automated. Therefore there must be some part of the original task remaining to be performed by the operator, which is the automation residual task. This is also true for "totally" automated tasks. These systems and tasks can create monitoring, fault detection, and management tasks (complementary automation tasks). Often these tasks are as complicated in nature as the original tasks.

An example might be a manual control first order tracking task. This task can easily be automated, but if there are changes in the conditions of the tracking problem then the operator must still monitor the situation. There the complementary automation task would be the monitoring task. We can see that the decision to allocate a function to the operator or to an automated system are not fundamentally the same. While a function can be totally allocated to the operator with minimum impact to the automated system the converse is not true. Note that the complementary monitoring tasks associated with system automation may have a detrimental effect on the system and operator.

4.3 Operational Considerations

Pilot objections can be categorized into two groups. One, that the system is not flexible enough, and two, that the system keeps the pilot from maintaining control of the aircraft (out of the loop). It is impossible and impractical for a system to be totally flexible. However if we look at flexibility as a goal, then we can design flexibility into systems to improve the acceptability and utility of automated systems.

One of the long-term problems which is caused by automation of control functions is the loss of the pi-

lots knowledge of the system operation. This is a condition which could impact system safety. The pilot must be operating within a manageable workload level and have a means of retaining sufficient control of the situation. The design must consider how pilots prefer to "stay in the loop"? Also how much flexibility can we put into the system and not make it nonstandard or too complicated.

Some functions have been automated because of superior performance of the automated systems. This may be a poor decision if the human performance is acceptable and the human needs to stay involved in this function to retain understanding of the situation. Automation modifies data flow, and could reduce the availability of key information. If operators cannot tell what is happening, they will be unable to keep their mental models of the system updated.

The solution to these problems is to develop automated systems which are in concept and implementation aids and limiters, not systems that do the flying. The propose of piloting aids is to reduce the workload on the pilot, not as low as is possible, but to a manageable level without any significant loss of the pilots situational awareness.

The idea of a systems limiter is to discourage the pilot from doing things which the automated system has calculated to be inappropriate, without taking command authority away from the pilot. An example would be a flight control system which increases the stick force as a pilot approaches a stall. The design of pilot aids and system limiters requires interactive design and testing with the pilot community. Automated designs require good systems engineering and integration. It is important to remember that the goal in aircraft systems design is to build the best possible system, and not to develop an elegant, simple design process. The use of pilot aids and limiters in lieu of fully automated systems requires higher pilot workload in many cases, but it is known that increasing workload is not always bad, as long as the workload is manageable. Most important however is that the pilot is not being relegated to the role of system monitor, but rather is staying involved with the dynamics of the aircraft or situation.

4.4 Failure Operation

Failure conditions have long been a major driver in the design of aircraft hardware systems. There are a number of major changes in the cockpit environment during systems failures and emergency operations. The new environment produced by an emergency condition can be characterized by a loss of planning and anticipation of coming events. The pilot is

forced into a purely reactive mode in a of rapidly developing situation and changing workload levels. The frequency of failures and emergency conditions may or may not warrant new systems dedicated to emergency conditions. However in the area of commercial aviation, avoiding one major incident could save hundreds of lives, and in the military, the number and consequence of failure/emergency conditions will rise dramatically when systems are introduced in a combat environment. Given those concerns, it is important that we investigate the possibility of redefining system allocation decisions based on the new ground rules associated with operation under failure mode. One possible solution would be to define an emergency mode of automation authority which could be manually engaged by the pilot.

5 Recommended Approach

Unlike many methodologies which attempt to assign functions to automation on man-machine task allocation, I will describe a concept for designing automated systems into an integrated cockpit where the pilot will control the system through use/monitoring of fully and partly automated systems.

There are different ways of looking at levels of automation. The traditional way of defining levels of automation is to look at how much authority is granted to the automated system. The level of automation is determined by how the automation function interacts with the operator, not the internal operation of the automation itself. The higher level automation has more delegated authority (McDanel 1988). In this section however I will define the automation implementation as a function of allocation phase and design environment.

1) Allocation during the design phase: During the design of a complex system such as an aircraft many of the decisions regarding the final system configuration are made. This is driven by the requirement to manufacture the aircraft. This is however changing with the development of computerised systems, multi-role aircraft, and multiple configurations of aircraft dependent on replaceable components. During this phase of system development allocation decisions are driven by the aircraft type, technologies available, and constraints placed on the system by other parts of the design that must be finalized early in the design process. It is important to understand the implications of the automation allocation decisions made at this phase of the design. Because of the nature of early design decisions it is nearly impossible to change them after the fact, therefore the allocation decisions

made during this phase will drive the interactions with other aircraft systems and will set the philosophy for the later allocation decisions. For example, an automatic climb out and altitude hold system is planned for a commercial airliner. This decision will drive present and future decisions with regards to the stall warning systems, fuel handing systems, center of gravity systems, automatic engine controls, and the associated pilot procedures. It is clear that decisions made during this phase can restrict the future development and operation of the system.

2) Allocation during the configuration phase: This phase of allocation has been under exploited. With the advent of interchangeable computerized avionics and cockpit systems the ability to utilize this phase has been greatly expanded. This phase is defined as the operational design the system in which the using organization configure the aircraft to meet the specific needs of its next mission, by installing and/or programming specific hardware components or subsystems. By shifting the allocation decisions from the the design phase to the configuration phase of the development we add flexibility to the possible application of the automated system. This phase of allocation decision can have a major effect on such areas as overall automation workload, and system(including the human operator) reliability and maintainability. If we can make enough of the allocation decisions at this phase we can add flexibility to system operation for the pilot.

3) Allocation during the operational phase: This phase of allocation is defined by actions that are taken by the pilot. There are a number of conflicting considerations involved in how the allocation decisions are made. These include, the amount of flexibility to allow a given pilot in making the allocation decisions. To effectively make these allocation decisions we need to consider which of the overall allocation decisions need to be made during each design phase of the project. The allocation decisions made during this phase of system definition are implemented during flight, but it is important to remember that they also impact training systems and the development and use of operational procedures.

Additionally automation decisions will be imposed by the type of aircraft cockpit design environment as follows. 1) Totally integrated cockpits/programs: There has been an understandable emphasis in the research and design community on developing fully integrated automation systems. These systems offer the most opportunities for dramatic changes in operation and their overall performance. The important concerns in this environment are maintaining levels pilot involvement and ability to attend to the mon-

itoring task. These concerns are accentuated by the inherent complexity of fully automated systems. Significant problems often arise from the very complex interactions between the various automated systems. The importance of using an integrated design team and project organisation is vital in working with the complex interactions, and the design implications of automated system design and implementation.

2) Partly integrated cockpits/programs: A partly integrated system is the most common environment. This can be defined typically as design enhancements. It deals primarily with how systems which are being designed meet current and near future needs. One area of the design in this environment which requires close examination is the provisions which are made for continued changes and improvements to the systems.

3) Insertion into already existing configurations (retrofit): As users of systems decide to keep older systems in service longer, there is an ever increasing need to insert automated systems into already existing designs. This implementation environment has its own problems. These problems can be grouped into two categories, one, integration with non-standard hardware and software, and two, changes needed to established operational procedures. The latter is easy to overlook but vitally important to the overall systems safety and training.

Each of these allocation phases and design environments has widely varying requirements for implementing automated systems. It is the intention of this paper to bring out these requirements so that the implications of automated systems implementation can be better appreciated.

6 CONCLUSIONS

The automation of systems and processes within the aerospace and other industries is taking place at an increasing pace.

The aerospace industry is concentrating too much on top down design and promises of automation. To deal with the ever increasing levels of complexity of automated system design, we need to address design on a less general and more individual basis.

The human interface and workload problems must continue to be addressed. The approach that should be taken is to maintain moderate workloads over as much of the operating environment as is possible. In the development of any system performance under failure condition is important, but in automated systems additional research and development work is needed to optimise the methodologies for design and evaluation of automated systems performance in fail-

ure modes. In the area of the human machine interface, it is clear that more work is needed in the area of defining the concept of mutual awareness. One of the issues that continually comes up in the evaluation of automated systems is the need for the operator to remain confident in the system functions. Mutual awareness is certainly a way of achieving this goal. With regard to the area of task allocation, it will be productive to re-evaluate automation task allocation methods taking into consideration the principles for residual automation and complementary automation tasks.

A team effort is necessary to deal with all the system issues in automation design. This will require a concentration on establishing realistic system performance based goals. It will be increasingly important for the design and implementation of automated systems that the effect on the entire life cycle of the system be considered, including how the system will be operated and modified over its lifetime. It is vitally important that the operational considerations of highly automated systems be researched.

Successes and failures of an automated system should be a larger consideration in a new system design. Design engineers need to look to operational experience lessons learned on human and machine limitations and interactions. It is much more practical to build on past successes and not over complicate designs just because new technology is available.

REFERENCES

1. Air Force Studies Board - National Research Council, "Automation in Combat Aircraft", Washington, D.C., National Academy Press, 1982
2. Chambers, A. B., Nagel, D. C., (1985, November). Pilots of the future: human or computer? Communications of the ACM, 28, 1187-1199
3. Fitts, P. M. (Ed.) (1951) "Human engineering for an effective air navigation and traffic control system." Washington, DC: National Research Council
4. McDaniel, J. W. "Rules for fighter cockpit automation" 1988
5. National Transportation Safety Board. "Scandinavian Airlines System DC-10, John F. Kennedy International Airport, Jamaica, New York, February 28, 1984", Report No. NTSB-ARR-84-15. Washington, D. C. 1984.
6. Price, H. (1985) The allocation of functions in systems. Human Factors, 27 33-46
7. Wiener, E. L. "Application of Vigilance Research Rare, Medium, or Well Done" Human Factors, 1987, 29(6), pages 725-736.
8. Wiener, E. L. "Human Factors of Advanced Technology ("Glass Cockpit") Transport Aircraft"

ACKNOWLEDGMENTS

The author wishes to express his thanks to Dr. McDaniel for his help and instruction in many areas of automation theory and practise. Special thanks to Ms Jan Gavern and Cynthia for there help, and support.

THE SHIP ENVIRONMENT SIMULATION PROBLEM

R.Thomas Galloway, Naval Training Systems Center*
 Frank E. Frey, Naval Training Systems Center*
 Dean Carico, Naval Air Test Center*

Abstract

Modeling the airwake and ship motion characteristics for real time simulation is a challenging task that must be done properly to be meaningful for the pilot in the simulator. Ship airwake models present a greater challenge than ship motion. This paper discusses applications to fixed wing carrier landing and VSTOL operations from amphibious and small deck ships. Improvement efforts should focus on developing standard modeling approaches for these ships to establish a basis for developing continuous improvements and uniform training potential. In addition, disk type rotorcraft models are too simplified to benefit from improved airwake models, full scale flow field data are needed for validation purposes, and quantitative flight test techniques are needed to document the aircraft/airwake interaction.

Introduction

As flight simulators increase in sophistication, so do user expectations of simulator capabilities. Current capabilities in increased, affordable computational power and realistic visual imagery provide good potential for improved fidelity in simulating high gain closed loop tasks such as flight operations aboard ships. Specific applications of ship environment simulations include training for fixed wing carrier landings, VSTOL operations from amphibious assault ships, and helicopter operations from destroyers and frigates with very small landing decks. Engineering development studies for Automatic Carrier Landing Systems and visual landing aids are also conducted in flight simulators. This wide variety of applications has resulted in a proliferation of modeling approaches with widely varying sophistication and fidelity. Simulation of the ship board landing task requires that several complex phenomenon be modeled and integrated properly; the major components being the aircraft flight dynamics, the visual scene content and field of view, minimal transport delays, ship motion, and ship airwake, or burble. Simulator motion cues are also beneficial for distinguishing between aircraft and ship movements and for identifying regions of turbulent air. Continuing obvious improvements are being made in all these components except the ship environment modeling area.

The typical structure of ship environment models in flight simulators is comprised of a module to compute ship motion dynamics and another module to compute ship burble. If more than one ship type is utilized in the simulator, then additional modules are implemented which contain data sets that characterize the motion and burble of each ship type.

Ship motion characteristics have been documented^{1,2,3,4} for a broad range of ship classes to help define operational limits for aircraft launch and recovery. This has resulted in a useful but complex data base that can be applied to real time simulation when properly analyzed and reformatted. In most training simulators, ship motion dynamics are modeled with three degrees of freedom (DOF). The three DOF are pitch, roll, and heave; in addition, the instructor has direct control of ship speed and heading. The pitch, roll, and heave DOF generally respond to ocean conditions represented in terms of sea state. Until recently, not much attention was devoted to the adequacy of these 3 DOF models and they were considered acceptable by default. A notable exception is the 5 DOF (roll, pitch, yaw, heave, sway) model developed for the SH-60B Operational Flight Trainer (OFT) where it was recognized early in the trainer development that the pilot must be presented with an accurate representation of the complex dynamics of small ships (destroyers and frigates) in high sea states. For fixed wing operations, the 3 DOF representation of large deck carriers was assumed adequate since their large size tends to attenuate the effects of sea states encountered in normal flight operations. However, analysis of recent fleet F-14 carrier landing accidents has revealed that the pilot's ability to maintain line-up is influenced significantly by the workload imposed by the yawing motion of his reference axis. That is, the pilot must cope with an easily excited Dutch roll mode when he tries too aggressively to follow the yawing motion of the landing deck centerline. Therefore, significant training benefits can be expected when the yaw DOF is added to the carrier motion drive algorithms. In summary, the typical flight training simulator models of ship motion provide only 3 DOF but fleet training requirements indicate that 4 and 5 DOF are necessary.

An assessment of ship environment simulation models is not complete without consideration of limitations imposed by other parts of a flight simulator.

*Member, AIAA

As mentioned earlier, the pilot's perception of the credibility of the ship environment is influenced significantly by the flight dynamics model, visual cues, motion cues, and transport delays. Pilot comments regarding the quality of a ship board landing simulation must be examined for problems in these areas before the ship motion and airwake models can be examined and analyzed in detail. The most common pilot complaint is visual field of view, especially in the downward direction for VSTOL and helicopter landing tasks. Dynamic response limitations imposed by excessive transport delays and inadequate aerodynamic modeling reduce the simulation credibility. Motion cues can result in improved performance in helicopter simulators, if properly implemented⁵. In summary, any efforts to provide high fidelity ship motion and airwake simulations will be wasted unless other major components - flight dynamics, visual cues, motion cues, and system dynamics (transport delays) are implemented properly first.

Flight Simulator Airwake Models

Ship airwake characteristics are far more complex and difficult to quantify as will be discussed for three major operational categories: fixed wing operations from large deck carriers; VSTOL operations from amphibious class ships; and helicopter operations from small deck ships.

Fixed Wing Operations from Large Deck Carriers

A naval aviator once said, "Landing [on a carrier] is the big equalizer. Miss a MiG and you can go back the next day. Foul up a landing and you really foul up." The Navy expends considerable resources in attempting to minimize landing "foul ups"; in fact, there are documented cases of aviators who failed "Carrier Qualification" but went on to careers as pilots in other services. The ability to "land on the boat" is a vital skill that must be maintained through use and practice.

There are several distinct phases to a carrier landing culminating in a traditional racetrack pattern that is always flown "left traffic" as shown in figure 1. It is what happens on the final approach leg that we are concerned with here; specifically, our interest lies in what happens during the last 25 to 30 seconds before touchdown. With a nominal 30 sec of flying time remaining, the aircraft is about 3/4 of a nautical mile from touchdown and at this point, a person on the carrier known as the "Landing Signal Officer" (or LSO) begins grading the approach in minute detail.

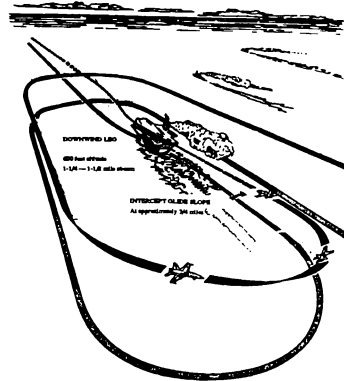


Figure 1. Carrier Landing Pattern

The LSO divides the approach into four stages and notes any deviations and pilot corrections during each stage. These steps are:

- The Start. Occurs with the aircraft about 4500 ft (3/4 nmi) and 30 sec from touchdown.
- At 1/2 nmi (3000 ft) from touchdown, the pilot is "in the middle"
- At 1/4 nmi (1500 ft) from touchdown, the pilot is "in close".
- The final stage is "at the ramp", as the aircraft crosses the deck edge.

The overall grade is based on deviations from the optimum approach path. Deviations are "major" or "minor" depending on the magnitude of the deviation and in what portion of the approach the deviation occurs. For example, the nominal hook-to-ramp clearance is 10.5 feet and crossing the ramp 5 feet below nominal is a major deviation. The airwake is an important factor because it affects the aircraft "in close" and has the potential to induce an altitude change of 5 feet or more. So it is important that pilots anticipate and correct for airwake effects.

Description of the burble. The airwake or burble is a change in the airflow caused by passage of the ship through the air. As with a truck or any other moving body, a low pressure region is formed aft of the ship. This low pressure area causes air flowing over the deck of the ship to be directed downward until it reaches the surface of the water. Then the air is deflected up to form a "roostertail". As the speed of the ship increases, the roostertail moves closer to the ship (figure 2).

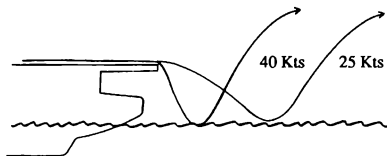


Figure 2. Simplified Airwake Profiles

Unfortunately, this simple description is not the whole story, so let's look at the major factors that complicate the picture:

1 - Obviously, the disturbance does not extend to infinity. Intuition tells us that the burble velocity components ought to be attenuated as we move laterally past the gunwales of the ship and as we move further axially behind the ship. There are no full scale data to describe the extent of these boundaries or tell us about the nature of the attenuation.

2 - The "notch" formed by the intersection of the angled deck and the straight deck adds rotation to the flow across the deck. Measurements behind the ramp⁷ on a Forrestal class carrier indicate that to the left of the centerline a region of updrafts exists, while to the right of centerline, a region of down drafts exists. It is possible that as the ship pitches, the air "dammed up" at "the notch" and flowing around the side of the ship "pours over" the flight deck. The process would be akin to a wave breaking over a sand bar. In any event, the phenomenon appears to vary from one class of carrier to another. Forrestal pilots noted a distinct "right wing down" tendency at the ramp; however, Nimitz pilots are unable to confirm this.

3 - The flow is not smooth — it is marked by turbulence. The level of turbulence will increase as the aircraft moves into the regions of the airwake disturbed by the carrier island and the aircraft parked on the deck. The location of this region of increased turbulence will change with the wind direction. For an axial wind (down the straight deck), theregion will be close to the ramp and will be relatively strong. For the wind directly down the angled deck, the aircraft will encounter an attenuated level of turbulence at a greater distance from the ramp.

4 - The influence of deck pitch angle on the flow over the deck. A carrier of the Forrestal class has an overall length of about 1000 ft and the center of pitch is located about 570 ft aft of the bow. Thus, when the deck pitches down 0.2 degree, the bow drops 2.5 feet and the

ramp gets elevated by 2 feet. It has been theorized that the pitching motion of the deck contributes to the rotary nature of the airwake; however, of more significance is the fact that the motion of the deck is responsible for a periodic variation of the flow.

When all of the aforementioned factors are combined, we see that the flow can be visualized as a "swirling roostertail" that varies with WOD magnitude and direction and deck angle and is spiked with varying levels of turbulence. The level of turbulence depends on your location in the burble relative to the island and any parked aircraft as well as the wind direction. The entire flow field is attenuated as a function of lateral displacement from the centerline and longitudinal displacement from the ramp.

The major impact of the airwake on fixed wing pilot technique seems to be mainly attributable to the steady state portion of the flow. For most fixed wing aircraft, it is necessary to anticipate a rise from the updraft portion of the "roostertail" followed by a settling as the aircraft transitions the downdraft region of the burble. Two power corrections are generally required: first a reduction to counter the rise and then an increase to counter the settle. Due to engine "spool-up" time, the key word here is "anticipate". If the pilot is already below glideslope and the stern of the carrier is pitching up, failure to anticipate settling from the airwake could result in a ramp strike.

So far we have looked at the nature of the airwake behind the carrier from a qualitative standpoint and examined the effects of airwake on pilots and airplanes. Now, let's review some of the efforts that have been made to quantitatively describe the airwake behind big deck carriers.

In the mid to late 1960's, research into carrier airwakes was done by Systems Technology, Inc (STI)^{6,7} and Oceanics, Inc^{8,9}. The results of these studies directly fed into airwake work at the Naval Air Development Center (NAVAIRDEVCECN). Eventually, the NAVAIRDEVCECN efforts¹⁰ and the first STI report⁶ were used to generate the "carrier landing disturbance model" included in MIL-F-8785C¹¹ and MIL-STD-1797¹².

The earliest Oceanics papers —and most of the following work — began with the idea that a three dimensional flow field is necessary to describe the air wake. The three dimensional velocity vector associated with each point in the flow field is typically resolved into the following components:

- a "longitudinal component" that is parallel to the centerline of the angled deck

- a "vertical component" that is parallel to the gravity vector
- a "lateral component" that is perpendicular to both the preceding components

Then STI postulated that each velocity component can have contributions from up to three sources:

- steady state contributions generated as a function of wind over deck magnitude and distance from the ramp
- contributions from the pitching motion of the ship's deck generated as a function of wind over deck magnitude, distance aft of the ramp, deck angle, and deck pitch rate
- random turbulence contributions generated using filtered white noise

In most cases, the final form of the model does not require computing nine elements: some of them are deleted for simplification. For example, the steady state contribution to the airwake is seen as being essentially two dimensional and the lateral steady state component is dropped.

In the recent past, models of airwakes in Navy aircrew simulators have been based on MIL-F-8785C and/or STI report 137-2'. This has been done primarily at the instigation of the Navy because these models represent a "culmination of learning" and were developed in conjunction with measured carrier airwake data. However, it must be noted that the airwake models based on these sources have not been completely satisfactory for aircrew training. The primary function of these source models is to provide a consistent environment for the evaluation of proposed aircraft designs and automatic carrier landing algorithms. Very detailed airwake models have been developed for these purposes but they usually lack the flexibility necessary to simulate a wide range of operating conditions required for training purposes. Compromises and simplifications made to "fit" carrier models into trainer computational systems have degraded their credibility. In order to examine some of the deficiencies and problems with airwake simulations in flight training simulators, let's look at three examples from the Navy inventory.

The first example is from the T-2C research simulator installed at the Naval Training Systems Center. This particular airwake model was adapted from an earlier F-4J training simulator model and it bears some resemblance to the STI model. Other T-2C training simulators lack a visual system and therefore have no carrier landing capability. The T-2C research simulator was used extensively for carrier landing

training experiments and Visual Landing Aids development work during the 1970's and 1980's. Although this airwake model has never been the subject of a rigorous evaluation, it has never been a source of pilot complaints. The T-2C airwake model computes velocity components only in the vertical and longitudinal directions. There is no lateral component. Also, there is no contribution from deck motion. The steady state contribution is generated from data tables and a random number generator provides the turbulent component. The routine is executed at 5 Hz.

The next airwake model is from the F/A-18 operational flight trainer (OFT). Here, the steady state contributions are generated with second order polynomials and contributions from deck motion and turbulence are computed in accordance with MIL-F-8785C. The lateral component is simplified by neglecting the steady state and deck motion contributions. In fact, lateral component of the burble is simply the longitudinal random component repeated. This model also executes at 5 Hz. In producing the airwake model for the F/A-18 trainer, the simulator contractor spent considerable time tuning the model based on pilot reaction. The result was a substantial increase in fidelity over the original airwake but that may be because the original model represented the airwake region as mere turbulence generated by random numbers. However, the pilots polled for this survey agreed that the burble region was too large and some of the effects too pronounced. In their words, "You're in the burble too long ... [and] ... there's too much sink at the ramp."

The last airwake model to be considered here is used on the USMC A-6E OFT. It also makes use of second order polynomials to generate the steady state contribution and, again, the random and deck motion components are computed per MIL-F-8785C. The lateral component of the airwake consists only of a turbulent contribution and this is computed by simply repeating the vertical random component. This model also executes at 5 Hz, so all in all, the A-6 and F/A-18 models are quite similar. Pilot reaction to the A-6E airwake model was somewhat harder to extract: Marine pilots don't spend much time "at the boat" and their ability to recall discrepancies is accordingly diminished. However, the consensus of opinion was that the simulator was more difficult to fly in the airwake than was the aircraft.

Table 1
Comparison of Airwake Envelopes

Aircraft Simulator	Longitudinal Distance Aft of Ramp — ft	Lateral Distance about Centerline — ft
T-2C	1573	+/- 128
F/A-18	5000	+/- 250
A-6E	2040	Unbounded

It is interesting to compare the envelopes for each of the airwake models (Table 1). The F/A-18 model is the one that most nearly agrees with our intuitive expectations; however, recall the pilot comments about the airwake region extending too far out ("you're in the airwake too long"). The T-2C airwake envelope looks impossibly thin, but this aspect was never evaluated directly. The pilots participating in the T-2C based experiments ranged in experience from students to high time LSO's and they flew both curved and straight-in approaches. No specific comments were received with respect to the airwake. The paucity of solid evidence derived from subjective pilot comments indicates a need for quantitative flight test techniques to document the airwake/airframe interaction. Data from such tests would be invaluable for validating airwake simulations.

All of the airwake models reviewed lacked features that were found to occur in the real world and in this sense alone, the models are deficient. Specifically:

1 - The steady state portion of the airwake does not move closer to the ship as the wind over deck (WOD) increases. While the steady state contribution does vary as a function of longitudinal displacement and while it is directly proportional to the WOD, there is no variation in the "roostertail" location.

2 - There is nothing that would explain (or cause) a rolling moment at the ramp. This phenomenon is apparently associated with Forrestal class ships'. Pilots whose experience was limited to Nimitz class carriers were not familiar with the problem. In any event, the existing airwake models do not predict "rolling at the ramp" for any class of ship.

3 - There is no variation of turbulence as a function of position in the airwake. Variations in the level of turbulence are caused by the carrier island and aircraft parked on the flight deck.

4 - There is no variation in air wake characteristics

with surface wind incidence angle. Normally, the ship is directed so that any surface wind blows down the angled deck. However, other conditions may have to be accepted if the ship has to maintain a given course. Also, the anemometers used on board ships may have errors on the order of two to three degrees or more. One effect of changing the angle of the surface wind from optimum to "axial" (the wind is directed down the straight deck) is that the turbulence from the island and parked aircraft will affect the approaching aircraft at a point closer to the ship.

VSTOL Operations From Ships

Rotary wing aircraft operate from a wide variety of ships ranging from large deck carriers (CV), to smaller deck carriers for amphibious assault operations (LHA, LHD, LPH), to much smaller aviation capable ships such as destroyers (DD) and frigates (FFG). In addition, fixed wing AV-8B VSTOL aircraft are deployed from several types of ships, mostly in the amphibious operations class. Flight operations with VSTOL aircraft are not as rigidly structured as fixed wing carrier operations due to the inherent flight path flexibility of VSTOL aircraft. However, the broad range of possible approach paths and the variety of ship profiles present a formidable simulation problem. The airwake here has the same elements as for a large deck carrier, but several significant differences exist:

- Approach flight paths are varied
- Ship motion is more significant due to smaller ship size
- More exposure to airwake due to lower approach speeds
- Significant turbulence and blanking effects due to small deck size and landing spot proximity to superstructure features
- Interference created by aircraft turning on adjacent spots

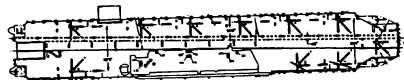


Figure 3
Amphibious (LHA) Landing Deck

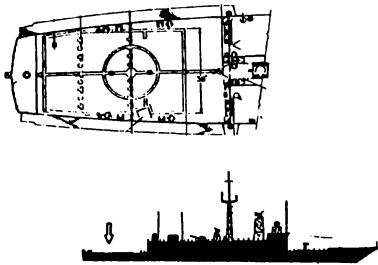


Figure 4
Small Deck Ship (FFG) Landing Area
Stern Approach Path

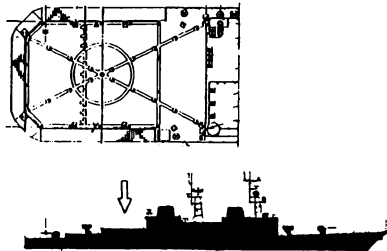


Figure 5
Small Deck Ship (DD) Landing Area
30 Degree Approach Path

A typical landing approach commences in a landing configuration in cruise flight similar to fixed wing carrier aircraft. Approaches to amphibious class ships (figure 3) are made at a 45 degree angle to the ship centerline toward an assigned landing spot on the deck. Approaches to small deck ships are flown either directly from astern (figure 4) or at an angle, typically 30 degrees (figure 5) to the landing deck on the aft end of the ship. A descending, decelerating, constant glide slope angle type approach is employed. Prior to landing, a transition to hovering flight based on visual reference to a moving platform is made. There are no automatic approaches, cockpit instrument glide slope indicators, or heads-up displays (except in the AV-8B), and the ship lighting package only provides rudimentary

glide slope and approach references. The smaller ships generally exhibit significant motion about all six degrees of freedom. Personnel on deck act as landing signal directors to advise the pilot on positioning and estimating lull periods in the ship's motion. Depending on the flying qualities and size of a particular aircraft, it may be held in a hover either just short of the ship or actually over the flight deck. This position is maintained until a quiescent period approaches, at which time the landing is commenced. Vertical landing is required within the confines of a designated spot on the deck, and once the decision to land is made, the maneuver is made expeditiously. Exact positional control must be maintained from initial positioning in the landing area until on deck. The SH-60B helicopter is assisted in this final phase of the landing task by a haul down system referred to as RAST (for Recovery Assist, Secure, and Traverse) installed in the landing decks of certain FFG, CG, and DD class ships.

Day Launch and Recovery Limitations

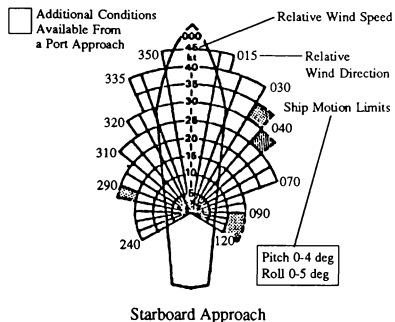


Figure 6
Typical Launch/Recovery Envelope

The compatibility of specific VSTOL aircraft and ship combinations must be established by actual at-sea trials. Static interface tests determine space and servicing capabilities. Dynamic interface tests are required to determine operational flight envelope parameters, and these tests are conducted by NATC. The cumulative effect of factors such as ship motion, ship-generated turbulence, obstructions, visual landing aids, field of view, and wind over deck establish the test environment. Aircraft performance and flying qualities

are then evaluated in this environment to establish actual takeoff and landing limitations. Test results are published for operational use as launch/recovery envelopes expressed in terms of relative wind direction and velocity for a specified level of ship motion (figure 6). The difficult problems of dynamic interface testing and associated attempts to develop flight simulator support are the subjects of on-going dialogs among technical experts representing the aircraft, ship, and fleet operator communities^{13,14,15}.

Simulation of Amphibious Ship Airwakes

There is no body of data to describe the flow field around any of the amphibious class ships. Therefore, a typical training simulator model of the amphibious ship airwake might be derived directly from the carrier burble model described above for fixed wing applications. That is, the burble model consists of three components: steady state flow effects, periodic effects of ship pitching motion, and random turbulence within the region of disturbed air. The airwake region is generally defined as a rectangular planform that is skewed with respect to the ship relative wind, while the vertical limit is defined with respect to glideslope height. This approach applies some sort of scaling to account for the difference in ship sizes relative to the reference Forrestal data⁷. This is an estimation at best and it can only be considered representative for winds a few degrees off the bow. Implementation of these characteristics in the AV-8B training simulators does provide representative cues for flight over the stern or bow for these headwind conditions. Modeling of airwake characteristics for beam or stern wind over deck conditions is somewhat arbitrary. The significance of these modeling weaknesses is that in the real world of amphibious ship flight operations, pilots report that their workload is much higher when landing with beam wind conditions than with bow winds. No specific quantitative data are available to guide the development of airwake models for these conditions. For the AV-8B simulator, "acceptance" testing was based solely on a brief qualitative test pilot evaluation. A positive note is that this model was still judged to be an improvement over the previous LHA airwake implementation, which utilized only random number generated turbulence. Amphibious ship airwake models in current assault/transport helicopter training simulators are not considered realistic by the evaluating pilots. Improvements to airwake models for these simulators are severely hampered by other factors such as the rotor aerodynamic model as will be discussed later.

The same data problem exists for the effect of the

ship's island or other protruding features in that it is either not modeled at all or is created in an arbitrary fashion that is not adequate for general training use. Airwake disturbances caused by the presence of the island in the flow field present some obvious significant workload effects on pilots when their approach path traverses this region. Arbitrary modeling approaches using geometric considerations only will provide some cue to the pilot but he cannot be assured of the exact location and strength of turbulent and blanked regions. In addition, NATC dynamic interface tests have revealed less obvious real world situations. For example, a port wind over deck condition significantly impacts a helicopter pilot's ability to land on a port side spot that is aft of the island because of winds reflected off the island. Under such conditions, some helicopters have insufficient control authority to counter the reflected wind which means the pilot cannot maintain his position over the designated landing spot. No existing simulator model is robust enough to replicate this situation, so no training can be provided to alert pilots to this phenomenon. Quantitative measurements of the flow field characteristics around amphibious class ships are needed, especially for relative wind conditions away from the bow and for island and superstructure effects under all wind conditions. Documentation of aircraft response to these conditions is needed in both quantitative and qualitative form to guide the refinement and validation of ship airwake models. The qualitative pilot ratings developed during NATC dynamic interface tests could be useful if applied to an orderly simulator model adjustment process. Quantitative test techniques are needed to document the aircraft/airwake interaction in an unambiguous manner.

Simulation of Small Deck Ship Airwakes

Prior to about 1982, Navy helicopter training simulators were not capable of replicating the small deck shipboard environment due to visual system limitations (scene content and field of view), and limited computational power which resulted in oversimplified models of both the helicopter dynamics and the ship motion and airwake. In addition, low computation rates and inefficient interface design induced large transport delays (more than 200 ms) which further caused the high gain shipboard landing task to be extremely difficult and unrepresentative in these simulators, even in zero wind conditions.

During the development of the training simulator for the SH-60B LAMPS Mk III helicopter, considerable attention was devoted to improving the simulation of ship motion and airwake models since the primary

mission of this helicopter involves operating in all sea states from the small FFG class ships. Reference data⁴ were identified which described the motion characteristics of these types of ships and this led to the development of 5 DOF ship dynamics models (the 5 DOF being: roll, pitch, yaw, heave, and sway) that are reasonable for most training scenarios. These ship motion models respond to conditions determined by wave front heading, sea state, and commanded speed and heading, and the ship motion exhibits the characteristic random quiescent periods where the pilot should commit to touchdown on the deck.

The airwake model developed for the SH-60B OFT represents the burble with three wind velocity components (x, y, z), each comprised of a steady state component and a turbulence component. This burble model operates at 15 Hz and is active only if the helicopter is aft of the landing deck and the WOD angle is less than 40 degrees. The burble components are stored in data tables which were obtained from a ship environment simulation model¹⁴ that was originally developed to support VSTOL aircraft development studies. The Vought model utilized wind tunnel data obtained from tests on a 1/50 scale model of an FF-1052 ship. A scale factor based on beam width was utilized to apply the data to DD-963 class ships. The SH-60B OFT treats the FF-1052 wind tunnel data as a generic data base and applies this same type of scale factor to compute burble components for either the DD-963 or the FFG-7 ship models. The generic data base is a region that is skewed with the ship relative wind that extends to 1000 ft downwind of the landing deck, and is 500 ft wide and approximately 100 ft high. The data tables contain non-dimensional parameters for wind components and turbulence level that are a function of distance from the ship reference point. These parameters are multiplied by WOD velocity, and the turbulence values are also multiplied by random number functions. All model component equations contain arbitrary gain adjustment terms. Fader terms are also utilized to ramp out effects when transiting a data base boundary region.

Initial development efforts with an experienced SH-60B test pilot appeared promising but subsequent fleet and test pilot evaluations have generated a consensus that while this airwake model may be better than previous models, it is still not very realistic. For example, some pilots still cannot discern wake effects from piloting difficulties induced by other simulator limitations, and other pilots have noticed that turbulence intensity does not appear to vary appropriately with range. The trainer is utilized to provide initial familiarization for Deck Landing Qualification (DLQ) and

RAST Landing Qualification (RLQ) flights but it is not considered adequate for safety workups by NATC test pilots prior to Dynamic Interface test deployments. An additional airwake model for a CG-47 ship was recently added to this simulator where the airwake components were derived primarily from geometric considerations. Pilot evaluations indicate that this model is equivalent to (ie, no worse than) the original models derived from wind tunnel data. It would appear that the wind tunnel data utilized here offered no significant advantage for this model implementation (in combination with the other simulator limiting factors). This same airwake modeling approach was utilized in a simulation study at NASA Ames¹⁷ for SH-2F helicopter landings on the DD-963. This study reported that the airwake effects were excessive and that air disturbances generated by the two stacks of the DD-963 were absent. In summary, the airwake models discussed here attempted to use the only airwake data available but pilot evaluations indicate that these models still do not provide representative characteristics for small deck ships. Shortcomings appear to exist in both the reference data and the model implementation. More work is needed to develop convincing real time airwake models that will not only represent small deck ships in general, but will include specific features of each class where appropriate. The nature of the airwake for small deck ships appears to be far more complex than for carriers and possibly the amphibious ships. Qualitative pilot ratings obtained during NATC dynamic interface tests would be useful for evaluating and adjusting potential models. As with the other ship classes, quantitative data from specific test techniques are needed to document the aircraft/airwake interaction in an unambiguous manner.

Some of the key aspects of the helicopter/ship interface simulation problem were discussed at length in a paper by Healey¹⁸ who has been studying this problem at the U. S. Naval Post Graduate School. This paper concludes that the freestream airflow to the ship and basic ship motion can be simulated relatively accurately, but better full scale measurements of ship motion and wind velocity are needed for validation purposes. Major shortcomings exist for simulating ship airwake and further work is needed for modeling helicopter dynamic response in flow fields with high velocity gradients and turbulence. Concern over computational power requirements is also expressed for both the helicopter dynamics and the airwake models. Computational power is now less of a concern (at least for the aerodynamics portion) than it was in 1987. As the paper stated, however, the small deck ship airwake remains virtually unknown, previous attempts to analyze it were faulty, and implementation of a real time model of such a complex

phenomenon requires much careful analysis. The nature of the flowfield revealed by studies of bluff body aerodynamics applied to wind effects around buildings indicate that airwakes are complex and very dynamic phenomenon. Analytical studies based on computational fluid dynamics¹⁹ have been initiated but much more work is needed to establish the credibility of this approach. Sheared flow, vortex formation and shedding, and turbulence intensities are difficult to measure, let alone model. Healey states "...today's knowledge of the flow around a bluff body oscillating in a sheared turbulent layer is very sparse and a major effort is required to gain an understanding of the highly complex nature of the ship's airwake." This is clearly a situation where further research is necessary.

Any efforts to improve airwake models in helicopter simulators must also address the rotor aerodynamics models. All of the current military helicopter training simulators employ disk type rotor models. According to pilots, these simulators do not provide a fully credible representation of flight in turbulent air or airwakes. This is not surprising since the frequency spectrum of the turbulent airwake region is beyond the bandwidth of typical disk-type rotor models. Rapid angle of attack changes induced by turbulence and flow field velocity gradients are simply ignored because the blade flapping dynamics are not modeled. Therefore, any attempts to improve airwake models with these rotor models will be filtered out and not apparent to pilots. In addition, current simulators apply the airwake and freestream turbulence only at the center of gravity of the simulated aircraft, thus ignoring the velocity gradients actually imposed across the rotor disk. In order for future airwake model improvements to be effective for rotorcraft simulators, they will have to be implemented in conjunction with a blade element type of rotor model in order to properly sense the flow field gradients across the disk area and then have the dynamic range to provide the proper response.

Problem Summary

The ship airwake simulation problem needs to be addressed in three areas: reference data for model design and validation purposes, real time simulation models that are credible to pilots, and validation test methods that are based on quantitative test techniques. Effort in these three areas should be applied to the three general ship categories discussed above (large deck carrier, amphibious, and small deck).

Reference Data

Large deck carriers: A useful body of flow field

data exists which describes some of the fundamental characteristics. More data is needed to describe additional characteristics such as the effect of the island superstructure and other objects, and the effect of variations in deck size. Extensive full scale flow field measurements are needed for validation purposes.

Amphibious ships: No body of flow field data is available for real time simulation for this class of ships. Carrier data have been utilized by applying scale factors but this is limited to head wind conditions. Modeling data such as that generated by wind or water tunnel tests plus extensive full scale flow field validation data are needed.

Small deck ships: The only wind tunnel data available for one type of ship in this category has not proven to be useful. Related studies of bluff body aerodynamics indicate that considerably more work is needed to define the complex flow fields and then develop practical models for real time implementation. Full scale flow field measurements are needed for validation purposes.

Real time models

Large deck carriers: Several simulator models exist which represent the fundamental characteristics of the carrier airwake. However, differences exist between them due to disparate development efforts that result in inconsistent performance, reduced utility for general applications, and no growth potential. A baseline carrier airwake model should be identified and implemented on high fidelity engineering flight simulators so that subject matter experts can develop improvements. The baseline model would serve as a reference for training simulators, help to ensure consistent training capability for the carrier landing task, and provide a focal point for developing continuous improvements.

Amphibious ships: Airwake models for this class of ship are generated by arbitrary methods and they possess numerous obvious shortcomings such as lack of island superstructure effects. Improvements should be channeled in two efforts: one with a near term goal of developing a suitable reference model by qualitative means, and the other with a longer term goal of developing a better model based on quantitative analysis techniques. Establishing a qualitative reference model as soon as possible would provide a uniform basis for training simulators plus provide guidelines for development of the quantitatively based model.

Small deck ships: No credible models exist for this class of ships. The complex nature of the airflow

around these types of ships would indicate that a sophisticated modeling approach is required but research is required to define the flow characteristics for practical implementation in a real time simulator. Such research may take a long time so airwake model development for small deck ships should follow the same two pronged effort mentioned above: establish a reference model for qualitative development to improve existing simulators in the near term, and devote a separate longer term effort to developing a more exact quantitatively based airwake model.

Validation Test Methods

Airwake models (all applications): The implementation of any airwake model could be easily validated if full scale flow field data were available. The test procedure would simply traverse the airwake region and report the computed flow field parameters for comparison with the full scale data.

Fixed wing aircraft: Evaluations of fixed wing flight simulators for the carrier landing task now rely on the current test methods applied in real world flight tests. These generally consist of quantitative data obtained to document the aircraft flying qualities and performance characteristics in the landing configuration, and qualitative pilot Cooper-Harper ratings and narratives of the carrier landing task. No data are obtained to specifically document the aircraft/airwake interaction during the landing approach. Test techniques and data recording methods should be developed to quantify the aircraft/airwake interaction characteristics.

VSTOL aircraft: Evaluations for shipboard operations of the AV-8B and helicopters result in data similar to the fixed wing carrier tests in that aircraft flying qualities and performance data are recorded and qualitative pilot ratings are obtained. Dynamic interface evaluations utilize a specially developed pilot rating scale¹³. While the myriad of aircraft/ship combinations and the diversity of approach conditions presents a formidable challenge, some form of quantitative measurement should be developed to document aircraft/airwake interaction to provide validation data for at least a few key approach conditions.

Conclusions and Recommendations

This review has shown that ship environment modeling for flight simulators is a difficult problem that appears to suffer from lack of focus. Ship motion and ship airwake models have been developed for a variety of applications. Ship motion was not discussed in detail

here because some solutions are available and the ship airwake is a far more difficult problem. Ship airwake modeling is difficult because of its complexity and a lack of flow field documentation capability. Improved airwake models are needed to realize the full potential of the other components of modern flight simulators so that pilot training and aircraft development needs can be met.

A unified approach is recommended where baseline airwake models for three primary applications (fixed wing carrier landing, VSTOL amphibious ship, VSTOL small deck ship) should be implemented in high fidelity engineering simulators and subjected to orderly development efforts. The development effort should have near term goals of producing a standard fixed wing carrier airwake model based on current models and data, and temporary standard models for amphibious and small deck ships based on the best available qualitative information. These models would provide a common reference for consistent trainer applications and a basis for continuous improvements. A longer term development effort should be devoted to developing quantitatively based airwake models for the amphibious and small deck ships. This would include parallel programs to obtain full scale data to guide and validate modeling efforts.

Additional general findings of this analysis are: (1) current helicopter simulators with disk type rotor models are too simplified to benefit from improved airwake models; and for validation purposes: (2) full scale flow field measurements are needed for all ship types and (3) quantitative flight test techniques are needed to document the aircraft/airwake interaction.

References

1. Kaplan, P., Sargent, T. P., "Theoretical Study of the Motions of an Aircraft Carrier at Sea," Oceanics, Inc., Report 65-22, January 1965.
2. Kaplan, P., "A Study of the Prediction Techniques for Aircraft Carrier Motions at Sea," AIAA Paper No. 68-123, January 1969.
3. Bascom, W., "Waves and Beaches, The Dynamics of the Ocean Surface, 1964.
4. Baitis, A.E., Meyers, W.G., Applebee, T.R., "A Non-Aviation Flight Motion Data Base for the DD-963, CG-26, FF-1052, FFG-..., and the FF-1040 Ship Classes," DTNSRDC Report SP-738-01, December 1976.

5. Ricard, G.L., Parrish, R.V., Ashworth, B.R., Wells, M.D., "The Effects of Various Fidelity Factors on Simulated Helicopter Hover," NAVTRAEQUIPCEN Report IH-321, January 1981.
6. Weir, D.H., "Analysis and Comparison of Carrier Airwakes," Systems Technology, Inc., STI Technical Report No. 158-1, April 1966.
7. Durand, T.S., "Carrier Landing Analysis," Systems Technology, Inc., STI Technical Report No. 137-2, February 1967.
8. Lehman, A.F., "An Experimental Study of the Dynamics and Steady State Flow Disturbances Encountered by Aircraft During a Carrier Landing Approach," Oceanics, Inc., Report 64-16, September 1964.
9. Lehman, A.F., Kaplan, P., "Experimental Model Studies of the Dynamic Velocity Fluctuations Existing in the Air Wake of an Aircraft Carrier," Oceanics, Inc., Report 65-21, Parts I and II, March 1965.
10. Nave, R.L., "Development and Analysis of a CV-1 and a FF-1052 Ship Airwake Model," NAVAIRDEVCEEN Report NADC-78182-60, September 1978.
11. Anon., "Military Specification, Flying Qualities of Piloted Airplanes," MIL-F-8785C, November 1980.
12. Anon., "Military Standard, Flying Qualities for Piloted Aircraft," MIL-STD-1797(USAF), March 1987.
13. Carico, D., McCallum, K.E., Higman, J., "Dynamic Interface Flight Test and Simulation Limitations," Eleventh European Rotorcraft Forum, Paper No. 100, London, England, September 1985.
14. Anon., "Helicopter/Ship Dynamic Interface (DI) Working Group Meeting Proceedings," Naval Air Test Center, April 1988.
15. Anon., "Minutes of the First Aerodynamic Ship Interface Conference," Washington, D.C., November 1989.
16. Fortenbaugh, R.L., "Mathematical Models for the Aircraft Operational Environment of DD-963 Class Ships," Vought Report 2-55800/8R-3500, September 1978.
17. Paulk, C.H., Astill, D.L., Donley, S.T., "Simulation and Evaluation of the SH-2F Helicopter in a Shipboard Environment Using the Interchangeable Cab System," NASA TM-84387, August 1983.
18. Healey, J.V., "The Prospects for Simulating the Helicopter/Ship Interface," Naval Engineers Journal, March 1987.
19. Mahaffey, W.A., Smith, C.E., "Turbulent Ship Airwake Environment Prediction Methodology," CHAM of North America, Inc., Report 4031/19, March 1987.

VALIDATION AND VERIFICATION OF FLIGHT SIMULATORS THE FACTS AND FICTIONS

Brian P. Hampson, F.R.Ae.S
CAE Electronics, Montreal, Canada

Abstract

The Validation & Verification processes for commercial aircraft flight simulators has largely grown from the FAA's Advanced Simulation Plan, formulated in the 1970's. It is now the base used by many Regulatory Authorities throughout the world. Since its introduction both the FAA and these other Authorities have developed the evaluation process. Manufacturers of aircraft & of flight simulators have introduced new methods to prove compliancy with these criteria. This evolution has sometimes tended to ignore or misinterpret some of the original philosophy of the FAA's Plan.

This paper seeks to re-establish that philosophy and highlight some current practices which have become common but which demonstrate a poor understanding of that philosophy and introduce potential flaws in the system. Some suggestions to correct these flaws are offered for consideration.

Flight simulation is not an exact science. This basic truth will be agreed by almost everyone in the industry, except for the most diehard optimists, and is demonstrated by the oft repeated criticism voiced by line pilots - "I don't care if it does meet all the data, it just doesn't fly like the aircraft". Accepting the technology constraints of affordable visual systems and the inherent limitations of the motion systems, which ensure that the simulated flight deck remains attached to the ground, why should there be a difference between the theoretical and the practical? These differences are becoming fewer at an accelerating rate as technology improves and the reservations of pilots, in respect of their proficiency being measured on a device other than the aircraft, are decreasing. I would even go as far as to say that the initial training leads to a better commercial pilot if carried out on a simulator rather than on the aircraft. There is, however, still a vestige of doubt in the minds of many of us as to the correctness of the simulation and it is this doubt which I wish to address today.

Verification, I take to be the evidence that the computer-implemented simulation corresponds to the mathematical model in behaviors of interest. Validation is a demonstration that a computer-implemented simulation corresponds to objective reality in behaviors of interest within some quantitative criteria.¹ It is my view that much of the problem lies with Validation rather than Verification. Indeed so much of the validation, which takes place today, is rooted in the subjective that by definition it cannot be Validation.

The FAA in its admirable Advanced Simulation Plan, now encapsulated in AC 120-40B, sought to solve some of these issues and this pattern has been adopted, in various ways, by many other Regulatory Authorities. What is frequently forgotten or misunderstood is that the FAA is not seeking solely to validate simulator performance against the aircraft. What they are seeking to do in defining the validation tests is also to establish a benchmark from which any deterioration in the standard of simulation will become obvious and quantifiable during the device's operational life. It is a method of ensuring that, as the simulator becomes older or is subject to modification over the years, its performance will remain fairly constant. A secondary objective is that the re-approval of a simulator at each of its recurrent inspections may be carried out against a known set of criteria. It is thus protected from the inevitable problems associated with such re-approvals being carried out by different inspectors, each of whom might apply his own set of subjective criteria. It is also planned to reduce the time necessary to carry out the recurrent inspections whilst, at the same time, ensuring that a minimum number of areas are covered over the period of a year. This is not to say that the FAA ignore a comparison with the aircraft. Far from it. They use validation against traceable flown data as the basis for the original Approval Test Guide, which becomes the Master ATG for that simulator's recurrent evaluations. The scope of the tests, however, was never meant to be more than a sampling. Reference to the AC 120-40B Appendix 4 expands upon that concept. There the example of a letter of application for evaluation includes a statement, to be signed by the applicant, testifying that the simulator has been tested by 'our pilots' and found to conform in respect of configuration, systems, sub-systems, performance and flying qualities to those of the aircraft. By no stretch of the imagination can that declaration be made, honestly, simply by the running of the ATG validation and functional tests. In CAE Electronics, we have the benefit of using trained simulator evaluation pilots, whose task it is, in conjunction with the system design engineers, to compile a set of tests and run them on the simulator to ensure that the performance of the device matches that of the aircraft within acceptable tolerances. These tests take about 250 man-hours to devise and cannot be properly completed, I submit, in less than another 200-250 hours. Accordingly they add significantly to the cost of the finished product but in the absence of we carrying out such tests our customers would compensate for this lack of testing by increasing their own testing.

Let us assume that all the necessary tests are carried out and the simulator's performance is thoroughly examined by extensive functional and subjective testing in addition to the validation tests included in the ATG. Where does this lead us? Firstly, the checking can only be as good as the data provided by the aircraft manufacturer and his vendors. In the case of the major aircraft manufacturers their approach to this is varied. Several attempts have been made by various bodies over the years, notably IATA, to define minimum standards for the data required both to build and to check the performance of simulators but to date no aircraft manufacturer has been able to meet all these exacting requirements and the data provided in the majority of cases is that which has been generated for other purposes than for flight simulation. This results in data obtained to meet the certification requirements of the aircraft being utilized extensively for the simulator. Whilst accurate, this data tends to represent regimes of flight which are not always relevant to training and, hence, to commercial flight simulation. The flight simulator will spend much of its training life close to the ground, on approaches, landings and take-offs but this is an area which is not always well covered by aircraft certification flights. To gather data in this regime is expensive and potentially hazardous as well as being time consuming. As a result, extrapolations are made in order to provide close to the ground information and the effects of this can sometimes be seen in the transition from the ground to the flight models in many simulators.

If the large aircraft manufacturers have difficulty in justifying the cost of providing data just imagine the effect upon the small manufacturers, who have not been used to the idea of producing very much data for any purpose. We have been engaged over the last year with several manufacturers of small aircraft of the commuter or regional airline type and, without exception, we have had to engage in data gathering exercises of our own using both the aircraft and the services of specialist data gathering companies in order to provide the basic information required.

The problems of data don't only lie in the area of the aircraft manufacturers, but increasingly with the vendors of on-board avionic equipment who have difficulties providing data which will permit the simulation of their units. When data is provided it tends to be incomplete data often restricted to input and output signal information. One argument often put forward is that the actual aircraft unit should be used on the simulator and stimulated by these inputs. Unfortunately, this ignores the fact that without adequate information of what goes on inside the box it becomes very difficult to diagnose box behaviour and determine how to produce the effect of simulated signal failure malfunctions.

When flown data is provided the actual data relating to the aircraft is often accurately described but that of the ambient conditions are generally only recorded at the start and finish of the test. Wind characteristics throughout the

whole of a cross wind landing/take-off is a good example of this type of data. More fundamentally, however, are the recognized limitations of currently used measuring equipment and procedures. Tests carried out on aircraft which involve long time-histories are unlikely to produce results capable of falling within the defined simulator tolerances. Apart from the poor precision of the measuring equipment, the spread of results across a fleet of the same model of aircraft, resulting from a questionable aircraft C of G, estimated thrust, varying friction levels in components and poorly monitored environmental conditions makes the chances of meeting such levels of accuracy fairly remote. To base the simulation upon a set of data obtained from one aircraft might, within certain limitations, permit the simulator to be made to represent that one single aircraft, but if it should turn out to be unrepresentative of the whole fleet then the line pilots will be fairly critical. The best models, therefore, are produced by the aircraft manufacturers who sample across all available data and produce test results which average these performances.

Data is, I submit, the greatest problem facing the simulation industry at the present time and the points I have tried to make so far are by no means exhaustive of the deficiencies. I have addressed some areas of concern in respect of fixed wing aircraft but the issues of rotary wing data deficiencies are much more fundamental. It has been said that the data available for these types of aircraft are comparable to that which was available for fixed wing aircraft twenty years ago.

Technology is not being supported by the quality nor quantity of data necessary to take advantage of state of the art simulation. Unless some legislation is introduced to change this, there is not going to be any large change in the situation.

The solution, which is being increasingly discussed amongst interested parties, is for the certification process of the AIRCRAFT to require the production of a minimum standard of data for the flight simulator. In this way the costs of obtaining and producing the data can be included in the price for the aircraft and then the costs are shared over the total number of aircraft of that type built rather than just to the operators who wish to buy a simulator.

Now, I would like to go a step further and assume that all the data we would wish for is provided. The simulator is designed using this and all the ATG tests show total compliance with the aircraft performance but the pilots subjective handling comments return to the original complaint - it's not like the aircraft. How can this be so? A simple example can illustrate my point. We are attempting a check of a cross wind take-off. The software driver for the ATG puts in the same amount of aileron as did the pilot of the test aircraft when the data was gathered. Given the same wind and other ambient conditions and assuming the simulation model is correct the simulator will respond as did that aircraft so the simulator plots will match the aircraft's. Is this a fair

check? On the surface it would appear so but let us look a little more objectively at what the pilot was doing during the recording of the original test, on the aircraft. Whilst still on the ground he was attempting to put in aileron to keep his wings level. In doing so he was depending upon the reactions of the undercarriage oleo struts as well as the cross-wind forces. In the flight simulator ATG check we have no knowledge of the oleos performance, even though we have design data, thus if we reduce the situation to absurdity and simulate a rigid undercarriage there will be no effect upon the ATG test because the aileron input will be that which the pilot of the aircraft felt was necessary to resolve the oleo and cross-wind effects in the real world. When the simulator pilot attempts a manually flown take off with the same cross wind component he too will employ aileron to keep his wings level but with a rigid oleo strut he will achieve this with a different amount of aileron. This will not become evident until he gets airborne and the oleo modelling is no longer a factor to be considered but at this point the simulator will roll in a manner the aircraft did not, perhaps causing it to crash. This is a good example of why total concurrence with the ATG may not ensure compatible performance with the aircraft and there are many more instances of this sort. Another is the question of sequencing of the modules in what is known as the man in the loop functions. Changing the sequence so that, say, the update of the aircraft attitude is one frame ahead of the calculation of the surface position will certainly affect the simulator's handling characteristics but this may be transparent to the ATG results and the latency. This is an extreme example but a significant one in showing how the ATG may indicate all is correct but the pilot feels something is wrong. It was for this reason CAE were very cautious about the use of multi-processor distributed processing, where the control of the sequencing could not always be guaranteed to be the same for each and every pass.

Some other anomalies may arise as a result of invalid assumptions of aircraft performance. It is generally assumed that the simulator's correct response to large control inputs will guarantee its response to normal pilot inputs, which tend to be very small. This assumption justifies the use of non-operationally orientated test cases using large alpha and beta step releases. If one considers the aircraft's non-linear aero response and the control surface gearing, there must be some doubt as to the validity of the basic assumption. Again, not all aircraft data involving control column or wheel inputs, actually have accurate column or wheel position recorded. To overcome this the software drivers used in the automated ATG's position the simulator's control surface to that of the recorded aircraft's and then the effect upon the simulator's attitude resulting from that control surface displacement is plotted. The control column/wheel is then back-driven. In general terms this is an acceptable situation because the surface to control column/wheel gearing is checked by stand-alone testing and the whole loop is tested by the subjective handling tests carried out by the pilot. It does beg the question, however, about the fidelity of the yaw

Damper and Turn Co-ordinator simulation.

Another well known area of doubt about the relevance of some aircraft data is that of control forces measured on the ground. From my own experience I have seen force v position plots taken from the same aircraft with some eight hours gap between them. The differences are sufficient to put the result out of tolerance. I have said in the past, somewhat with tongue in cheek, that the application of the simulator ATG tolerances to the aircraft would frequently prevent it from being approved as a training device. Exploration of the differences shows that they arise from a variety of causes. Hot and cold soak of the aircraft produces significant change in the plots. The accurate positioning of the force measuring equipment is similarly critical. We, at CAE Electronics, once set up a flight control assembly in our test laboratory and selected three different engineers to install the measuring equipment and obtain a simple set of plots, each person removing the equipment at the end of his plotting. No two sets of results were the same and the differences ranged from minimal to being out of tolerance. When asked to repeat the exercise using the same rig and equipment later in the day a further three different plots were obtained with the same scatter. This has led us to the design of a new type of force measuring equipment to minimize differences arising from its installation. One is left with the question though, do the actual aircraft plots suffer from the same problem when, as is normally the case, strain gauges are not used? Consideration should also be given to the changes which arise between a set of plots taken on the ground and those obtained in flight, body flexing, temperature effects upon cable stretch, cable pulleys, grease, seals, etc., all have some effects upon the resulting plots and may account for some handling differences noted by the line pilot. That these differences exist in the aircraft is not considered to be relevant to the simulator. We all take the aircraft for granted and, so long as it feels in the right ballpark, the handling pilot rarely questions the undoubted differences between this aircraft and the one he last flew. In the simulator we all become hyper-sensitive to real or imagined variances. This leads me to another observation associated with validation/verification by subjective checking. When we criticize the simulator's performance in a certain manoeuvre or drill, on what are we basing our comments? Have we actually experienced that particular situation in the aircraft and particularly noted it with some amount of objectivity or have we based our experience upon some experience passed to us third hand or, even worse, remembered from another simulator? I flew the B747 for about ten years but rarely experienced any of the emergencies which I used to evaluate when accepting simulators of that type.

Similarly using thrust ramps to provide the engine powers for testing of the aerodynamic properties, leaves large portions of the engine simulation virtually untested. This latter case becomes even more of a problem when one considers the simulation of engines with FADEC types of control. What parameter can one use in this case, when the thrust levers may be giving little

indication of what is happening to the engines, coupled with the fact that the necessary data is incomplete for an analysis and there is poor recording of the ambient conditions which the FADEC was using during the aircraft testing? Perhaps there is no real alternative to the use of thrust ramps but these need to be supplemented by further checks of the engine simulation or be restricted to those being produced by the installed engine model of the simulator. Even here, there are going to be problems because aircraft manufacturers and their test pilots are unlikely to risk the aircraft or their necks to run real-life tests of an engine cut at Vmcg. We are inevitably, in regimes such as this, going to have to accept engineering simulator/predicted data.

Even when verification is being accomplished by the comparison of simulator performance to aircraft manufacturers engineering simulation data (i.e. Predicted Data) it is frequently felt that such verification, being computer to computer comparisons, should be done with zero tolerance. What is not realised is that rarely is the predicted data derived by running totally integrated models in the engineering computer. In most cases each model is run independently and the results fed into the next. This is not a good analogue of the flight simulator where the models all have to be integrated and run in real time and where significantly different results may be obtained. Again, as discussed earlier, the sequencing of the modules may have an effect which is not noticeable until subjective handling checks are carried out.

Perhaps a more fundamental consideration is whether we are correct to even attempt to make the flight simulator a replication of the aircraft. If the device is to be used for research or as a method of proving aircraft performance, then the answer is self evident. It is a questionable aim through if the device is meant to be a training tool. What we should be concerned about here is whether the simulator training transfers skills effectively to the aircraft. Current thinking was established when the majority of flight training was done in the aircraft. It is only recently, and then only in some countries, that the necessity of a flight simulator in a training program has become obligatory either for reasons of economics or by legislation. In the absence of any studies to the contrary we have developed the flight simulator to be a replica of the aircraft rationalizing that in this way we have a one to one compliance with the training obtainable from the aircraft. What is conveniently ignored is that it is known that the aircraft itself, for many reasons, is not a good vehicle for the transfer of training. We then compound this by accepting that in some areas, notably motion and visual simulation, there are inherent degradations which cannot be overcome with current technology. In short we poorly replicate something which is a poor transfer of training vehicle.

It is all very well being a critic of the current situation, listing and describing all of its failings but that is not very constructive. I shall close my presentation therefore by trying to suggest some means by which the art of simulation might become more of a science. These are: -

1. An improvement in the definition of and regulatory action in respect of, the data requirements for flight simulation. Preferably, this should be actioned upon an international basis. Such is one aim of the Royal Aeronautical Society of the U.K, which has held two Seminars and convened a Working Group, consisting of representatives of the major interested parties, to consider the recommending of a set of International Criteria for the Evaluation and Approval of Commercial Flight Simulators. The initial response has indicated a recognized need from all parts of the world and the time is obviously ripe for this initiative.
2. Greater emphasis to be put upon the need for comprehensive validation and verification testing as an adjunct to the limited ATG testing now typically carried out.
3. Recognition that ATG tests are to establish a baseline for recurrent checking and not primarily a method of proving compliance between the aircraft and the simulator which will permit a reduction in the wider aspects of testing.
4. Initiation of research into the effectiveness of the flight simulator as a training tool and the assumption that replication of the aircraft is what is required.
5. Consideration as to whether validation and verification of the flight simulator for the purposes of matching its performance to that of the aircraft should be considered separately from the need to verify that the simulator is being maintained to a standard equivalent to its initially evaluated status. This might entail a more thorough initial evaluation but a shortened and, perhaps, totally automated recurrent evaluation coupled with stringent configuration controls.

1. Dr. James H. Bradley "Glossary of Terms
Used in Simulators"

ANALYZING TIME DELAYS IN A FLIGHT SIMULATION ENVIRONMENT

R. E. McFarland*

NASA Ames Research Center, Moffett Field, California
and

J. W. Bunnell**

Syre, Inc., Ames Research Center, Moffett Field, California

Abstract

Transport delay in a multirate flight simulation environment is examined. An equivalent systems model is developed that quantifies the contributions of individual components and their sampled-data interactions. Mathematical algorithms used in the discrete implementation are also considered, because they are important elements of a flight simulation system. The equivalent systems model was used to demonstrate the consistency and accuracy of data obtained in the flight simulation facility at Ames Research Center. It showed that effective time delays in simulation models, including delays in scene presentation to the pilot, are considerably less than might be assumed by casual examination of raw data obtained from component-level experiments.

Introduction

A total simulation system is rarely exercised in obtaining time delay measurements. Notably, compensating algorithms are usually disabled, and drive signals used in an investigation often bear little resemblance to pilot inputs. In Ref. 1 (p. 14), for example, a step signal was sent through an analog-to-digital converter (A/D) to produce a discrete change in a computer generated image (CGI display). The difference between the time of the command and time of the response purportedly measured the system delay. Such an experiment has several features which require interpretation, in order to apply the results to the flight simulation environment.

* Aerospace Engineer.

** Simulation Programs Manager. Member AIAA.

Copyright ©1990 by the American Institute of Aeronautics and Astronautics, Inc. No copyright is asserted in the United States under Title 17, U.S. Code. The U.S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Governmental purposes. All other rights are reserved by the copyright owner.

First, during the A/D conversion process, a step input has a uniform distribution within a sample interval. The signal also has an approximately uniform distribution within the CGI computer, where compensating logic (which usually accounts for asynchronous data transmittal between computers) cannot utilize discontinuous data. The resulting compound distribution contributes as much as two computer cycles of delay. This constitutes a major difference from the actual simulation environment, where inputs are generally sampled such that significant activity does not occur between sample intervals—a fundamental hypothesis of real-time simulation. If the hypothesis fails, then the simulation fails.

Second, in Ref. 1 specifically, since the signal was not integrated, it could not be predicted to the end of the cycle, a function normally handled by an integration process, as will be discussed. This introduced an entire cycle of delay.

Third, recently developed CGI compensation logic, used in flight simulation at Ames Research Center, requires that CGI commands also be representative samples of continuous signals. Although the compensation scheme was developed subsequent to the experiment of Ref. 1, its availability at that time would not have influenced results, because of the continuity requirement. Hence, a repeat of this particular experiment today would still manifest the CGI pipeline delay. However, because of the compensation algorithm, this delay is not observed in flight simulation (over the frequency range pertinent to handling qualities research).

Thus, the data of Ref. 1, although accurate, are nonetheless misleading because they indicate a delay on the order of 150 msec, whereas in the flight simulation environment at Ames the effective delay is more like 15 msec.

A model of flight simulation systems is therefore needed that quantifies the contributions of various delay components and contains the proper algorithmic relationships. For this reason an equivalent systems model (ESM) has been developed. It reconstructs complete data paths

in the simulation environment, where both algorithms and sampled-data effects are important elements. To establish a foundation for this material, the sampled-data phenomenon is examined.

From the perspective of a discrete, multirate, man-in-the-loop flight simulation, the sampled-data system discussed in this paper is outlined in Fig. 1. This figure illustrates the sampled-data elements of a simulation that are pertinent to the creation of a discrete realization from a continuous system.

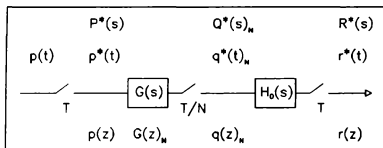


Fig. 1. Sampled-data system.

$G(s)$ represents the multirate simulation model, sampled N times for each input-output (I/O) cycle time T . Because this is a sampled system, the function $G(s)$ consists of the product of a model transfer function $P(s)$ and a data-hold function $H_0(s)$, as required in a data reconstruction process. This will be discussed further in the section "The Discrete Multirate Model." $G(s)$ may, of course, consist of a network of transfer functions, with some resultant equivalent data-hold function. The symbol $p(t)$ represents a pilot input, and $q^*(t)_N$ is a model output at the multirate level of computation. The final output $r^*(t)$ is sampled at the same rate as the pilot input (the I/O rate). This is accomplished in the ESM via a zero-order hold (low-pass filter) shown as $H_0(s)$ in Fig. 1.

Using the above outline, an ESM is developed for the analysis of delays in flight simulation. Models of multirate computer processes using z -transform notation are treated first, using essential elements from sampled-data theory.

Input Sampler

Pilot control inputs are represented in Fig. 1 by the time function $p(t)$, here assumed continuous with Laplace transform $P(s)$. Neglecting, for the moment, converter dynamics and possible prefiltering, the A/D operation consists of a switch that samples the input every T seconds and delivers a pulse train of data to the digital computer. Since the transfer function of a linear system is the Laplace transform of the impulsive response of the system, at the sample points the pulse transfer function is defined by the ordinary Laplace transform.

"The pulse transfer function relates only the output pulse sequence to the input pulse sequence. It does not relate the continuous output to the pulse sequence at the input" (Ref. 2, p. 69). As an example of the sampling process, consider the input sine function,

$$p(t) = \sin(\omega t)$$

Its pulse transfer function is obtained from any table of Laplace transforms,

$$P(s) = \omega / (s^2 + \omega^2)$$

and is defined as producing a series of impulses represented by the z -transform

$$p(z) = Z\{P(s)\} = \frac{\sin(\omega T)}{z - 2 \cos(\omega T) + z^{-1}}$$

"When a unit-impulse function is applied to a linear system with transfer function $P(s)$, the output of the system is simply the impulse response of the system" (Ref. 3, p. 74). To examine the equivalent pulse train within a digital computer, the z -transform is reduced to a difference equation using a difference operator for powers of z ,

$$y_{k+1} = 2 \cos(\omega T) y_k - y_{k-1} + \sin(\omega T) p_k$$

With quiescent initial conditions and a unit impulse for an input, a trivial induction shows that at the sample points, for all $k \geq 0$,

$$y_k = \sin(\omega k T)$$

Thus, in the process of sampling continuous data, the sampler (perfect A/D converter) does not produce signal delay.

Interpolation

Consider the multirate system shown in Fig. 1, in which N is assumed to be an integer. As will be shown in the section "The Discrete Multirate Model," the output of a multirate system may be related to its input using z -transforms.

When the new sampling rate is higher than the original sampling rate the process is called interpolation, because samples of the original physical process are collated from a reduced set of samples. In Fig. 1 the signal $q^*(t)_N$ is sampled at a rate N times faster than the sampling rate at the input. If $P^*(s)$ is defined as the pulse transform of $p(t)$, the Laplace transform of the intermediate result sampled at the expanded rate, $Q^*(s)_N$, is given by (Ref. 2, pp. 221-222; Ref. 3, pp. 82-85, 353-357)

$$Q^*(s)_N = \sum_{n=0}^{N-1} [G(s) e^{j\omega n T / N}]^* P^*(s) e^{-j\omega n T / N}$$

The z -transform of this expression is given by

$$\begin{aligned} q(z)_N &= \sum_{n=0}^{N-1} Z \left\{ G(s) e^{sTn/N} \right\} p(z) z^{-n/N} \\ &= \sum_{n=0}^{N-1} \sum_{k \geq 0} g[(k + n/N)T] e^{-(k+n/N)Ts} p(z) \\ &= \sum_{m \geq 0} g(mT/N) e^{-mTs/N} p(z) \\ &= G(z)_N p(z) \end{aligned}$$

where the development includes the substitution of $m/N = k + n/N$ (Ref. 3, p. 357). Hence, $G(z)_N$ can be obtained from the ordinary z -transform $G(z)$ by the replacements

$$G(z)_N = [G(z)]_{\substack{z = z^{1/N} \\ T = T/N}}$$

Thus, the multirate z -transform of the output of a multirate sampled system is obtained from the ordinary z -transform of the input and the multirate z -transform of the model (with its associated data hold). In terms of a simulation model, therefore, the z -transform representation of one subcycle of the model is sufficient for an analysis of a multirate system, providing the decimated output can be obtained from the multirate output.

Because of this relationship, it is natural to define the multirate cycle time as $\tau = T/N$, where τ represents one subcycle of the model. The z -transform operator is given by

$$z = \cos(\omega T) + j \sin(\omega T) = \cos(N\omega\tau) + j \sin(N\omega\tau)$$

Multirate Signals

The signal $q(z)_N$ represents data at the expanded sample rate. Because of its extended Nyquist limit, $f_N = N/2T$, it contains frequency content beyond the Nyquist limit imposed by the final (output) sampler, $f_T = 1/2T$. As will be shown in the next section, the baseband of interest ($f < f_T$) is recovered in the ESM via a low-pass filter. This filter may be a data hold.

Without the low-pass filter, the spectrum of $q(z)_N$ would contain not only the baseband frequencies of interest (i.e., $f < f_T = 1/2T$) at the I/O level of computation, but also images of the baseband centered at harmonics of the original sampling frequency. This extended frequency content would occur at all multiples of the original sampling frequency, limited only by the expanded Nyquist frequency, $Nf_T = f_N$, beyond which nothing may be observed from a discrete model.

In a real-time multirate simulation environment, a nonlinear model may also generate frequencies beyond f_T (up to f_N). Indeed, the reason for implementing a multirate

model is usually the high-frequency content generated within a model by internal nonlinearities and feedback paths (which require algorithmic approximations in the convolution to a discrete realization). Hence, it is sometimes observed that a smaller effective cycle time (than the I/O cycle time) is required to attain numerical stability.

The consequences of multirate processes, however, deserve some discussion. If a smaller internal cycle time is required, then the discrete model may produce frequency activity beyond f_T . If these frequencies are present in the outputs of multirate components, then upon decimation to the I/O rate, these signals will alias into the I/O bandwidth and will appear at some other frequency location; this will alter, and possibly destroy, the integrity of the model. This phenomenon was reported in Ref. 4, wherein an "aliasing equation" was provided to show that aliased frequencies may adversely influence the pilot/simulator bandwidths, especially in rotorcraft models.

Good reason exists for implementing a multirate model, providing the contaminating frequencies are filtered at the expanded rate prior to decimation to the I/O rate (see Refs. 4 and 5). In that case, upon decimation, only attenuated signals are aliased into the I/O passband. Filtering algorithms require special attention, however, because they must not adversely influence either phase or gain in the pilot-vehicle bandwidth.

The multirate technique without filtering is often inferior to other techniques for attaining stability. Although it may be attained, the original problem is exchanged for other problems, e.g., aliasing. In certain instances there are alternate solutions for attaining stability; these include algorithmic substitutions, purging the model of components that create frequency content beyond the I/O bandwidth, and anti-aliasing filters in the model.

The spectrum of $q(z)_N$ is thus of interest because activity beyond f_T influences decimated simulation outputs $r(z)$ via the aliasing phenomenon. However, signals that originate in a model beyond the I/O Nyquist frequency cannot be aliased by the ESM. Fortunately, this is of little consequence in the determination of effective time delay, which involves evaluations only at low frequencies.

The Discrete Multirate Model

The behavior of the input data between sample points is assumed by selecting a data hold. The discrete model $G(z)$ is obtained from the data-hold assumption (or a combination of data holds). The data hold serves as a low-pass filter, which is required in extracting the signal spectrum between sample points. A temporal shift may also accompany a data hold. Three holds are of interest here; they may be obtained from the Newton-Gregory formula (Ref. 2, pp. 31-39). They are given by

$$H_0(s) = \frac{1 - e^{-sT}}{s}$$

$$H_1(s) = \left(\frac{1 - e^{-sT}}{s} \right)^2 \left(\frac{1 + sT}{T} \right)$$

$$H_2(s) = \left(\frac{1 - e^{-sT}}{s} \right)^2 \frac{e^{sT}}{T}$$

$H_0(s)$ is the zero-order hold, in which the input is assumed to remain constant between sample points. It tends to advance the output by a half cycle from the most recent input point, although it purports to advance outputs by a full cycle. Its utility is often overstated because it happens to produce perfect answers (advanced one cycle) for step inputs. $H_1(s)$ is the first-order hold, in which the input is linearly extrapolated beyond the most recent input point. It tends to advance the output by a full cycle. $H_2(s)$ is the triangular hold, in which the input is linearly interpolated during the interval (Ref. 2, p. 288). It delivers an output that is concurrent with the most recent input.

The multirate discrete transfer function is obtained from

$$G_i(z)_N = Z\{H_i(s)F(s)\}_N$$

where the subscript i is used to keep track of the selected data hold. At the multirate level of computation, $G_i(z)_N$ relates an output to an input. It is important to keep in mind that the selection of the zero-order or first-order data hold produces advanced outputs (a time shift).

All three data holds are useful in creating a $G(z)$ that describes models containing networks of transfer functions that include feedback paths. Whenever a time advance (shift) is not required, the use of the triangular hold is highly recommended. The zero-order hold is not recommended unless phase is immaterial (as in the case of random inputs), or unless it is needed for stability in a feedback loop, in which a first-order hold implementation could produce a negative gain margin (unstable discrete realization).

Because $G_i(z)$ is transformed to $G_i(z)_N$ in the ESM, the data hold is effective over the extended Nyquist limit. Hence, $G_i(z)_N$ tends to match the spectrum of $F(s)$ beyond the I/O Nyquist limit. Because of decimation, however, a different spectrum is observed upon output.

Decimation

The spectrum created from the interpolation process beyond the I/O Nyquist limit must be purged from the mathematical representation prior to observation of the decimated output $\tau(z)$. "Based upon well known sampling theory, in order to lower the sampling rate and to avoid aliasing at this lower rate, it is necessary to filter the signal with a digital low-pass filter which approximates the ideal characteristic" (Ref. 6, pp. 302-303):

$$L(j\omega) = \begin{cases} 1/N & \omega \leq \pi/T \\ 0 & \omega > \pi/T \end{cases}$$

A filter that reasonably attains this characteristic is given by

$$L(z) = \frac{z^{-1}}{N} \sum_{n=1}^N z^{nN} = \frac{1 - z^{-1}}{N(1 - z^{-1/N})}$$

a term we call the rate conversion factor. It converts spectra at the expanded rate to spectra at the I/O rate, and is thus an I/O passband filter. The gain of the rate conversion function $L(z)$,

$$|L(z)| = N^{-1} |\sin(\omega T/2) / \sin(\omega T/2N)|$$

is plotted in Fig. 2 for $N = 2, 3, 4$, and 5. The abscissa of Fig. 2 represents the normalized frequency, with a value of unity being equivalent to the multirate Nyquist frequency (π/τ).

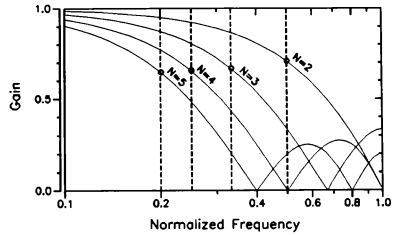


Fig. 2. Conversion gain.

The I/O Nyquist frequency after the decimation process (π/τ) is also indicated in Fig. 2; the gain at this frequency approaches $2/\pi$ as N increases. These curves illustrate that the rate conversion factor $L(z)$ does a satisfactory job in attenuating frequency content beyond the I/O Nyquist limit.

In order to relate the decimated output $\tau(z)$ to the input $p(z)$ we must also include the computer delay per subcycle, $z^{-1/N}$. The multirate portion of the ESM is then represented by the transfer function relationship

$$W(z) = \tau(z)/p(z) = z^{-1/N} L(z) G_i(z)_N$$

Comparisons

The accuracy of $W(z)$ to predict delay times in typical flight vehicle models can be demonstrated by comparing its results with those obtained using time-series analyses or real-time simulations. To accomplish this, the delays in simulation were established using Monte Carlo techniques,

with five random variables. A fifth-order Laplace set of functions was selected as being representative of flight simulation. This set is given by

$$F(s) = \frac{\omega_1^2 \omega_2^2}{(\lambda s + 1)(s^2 + 2\zeta_1 \omega_1 s + \omega_1^2)(s^2 + 2\zeta_2 \omega_2 s + \omega_2^2)}$$

The five random variables were given the ranges

$$2\pi < \omega_1 < 4\pi$$

$$4\pi < \omega_2 < 8\pi$$

$$0.1 < \zeta_1, \zeta_2 < 0.9$$

$$0.05 < \lambda < 0.5$$

The phase angles, and hence the equivalent time delays, can easily be computed at any given frequency by substituting $j\omega$ for s in the Laplace representations. Selecting the observation frequency as $\omega = \pi(1/2 \text{ Hz})$ produced 300 random time delays, as shown in Figs. 3(a), 3(b), and 3(c) by D_i . Values up to about 600 msec were observed using the selected range of random variables. Time delay is defined in this paper as the negative of the relative phase shift divided by the frequency.

The time-series outputs of the 300 functions were obtained using a cycle time of 30 msec. Multirate simulations were performed, in which the number of multiple loops (N) ranged from one to ten, in sequential simulations. The driving function, an input to the simulations at the I/O rate of 30 msec, was a 1/2-Hz sine wave (permitting comparison with the Laplace computations). The multirate portion of the simulation model was handled by an internal DO LOOP, within which the cycle time was $\tau = T/N$. This was computed for each of the data holds i discussed in this paper, by use of software developed from the techniques of Ref. 7.

The phase angle from each simulation was computed by trigonometric least-squares techniques, in which the phase angle (and resultant time delay) is identified. By differencing the observed time delay from that obtained from the Laplace functions, the relative time delay (produced from the discrete simulation) was computed. The results of these experiments are presented in Figs. 3(d), 3(e), and 3(f). This error is presented as ΔD_i for the various holds.

The first 10 runs of Fig. 3 used $N = 1$, the second 10 runs used $N = 2$, etc. For example, runs 91 through 100 used 10 multiple loops.

The results of this analysis show dependence only on the specific data hold (i) and on the number of multiple loops (N). The results are independent of the functionality (simulation model!) providing only that the equivalent phase shift caused by the discrete realization process (data hold) is isolated. From these data, and other data obtained at different cycle times, the delay produced by multirate simulations (with a reasonable bandwidth) is found to be, to negligible measurement error,

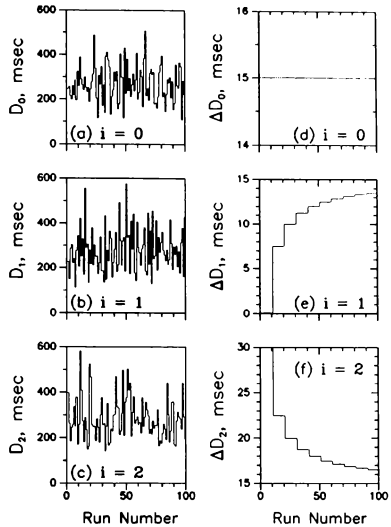


Fig. 3. Monte Carlo results.

$$\Delta D_0 = \frac{1}{2}T$$

$$\Delta D_1 = \frac{1}{2}T(1 - 1/N)$$

$$\Delta D_2 = \frac{1}{2}T(1 + 1/N)$$

whenever the simulation model uses a zero-order, first-order, or triangular hold. When the simulation model uses some other data hold or combination of data holds, the relative delay may also be computed, by first determining the equivalent data hold resultant from $G(z)_N$.

Returning to the ESM, we illustrate in Fig. 4 the discrete input-output relationship $W(z)$, which contains all of the elements necessary to approximate the time delay in a multirate simulation model. It accurately predicts all of the ΔD_i because of the factor $z^{-1/N}L(z)$, which by itself produces a time delay that is independent of the data hold i :

$$T_w = \frac{1}{2}T(1 + 1/N)$$

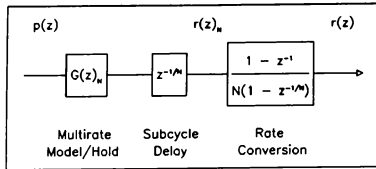


Fig. 4. Multirate portion of ESM.

Considering only the model with its associated data hold $G_i(z)_N$ at the multirate level of computation, it produces (at low frequencies) an advance of $T/2N$ for the zero-order formulation ($i = 0$), an advance of T/N for the first-order formulation ($i = 1$), and no change at all for the triangular formulation ($i = 2$). Since phases add, we see that $W(z)$ predicts the delays

$$\Delta D_0 = T_W - T/2N = \frac{1}{2}T$$

$$\Delta D_1 = T_W - T/N = \frac{1}{2}T(1 - 1/N)$$

$$\Delta D_2 = T_W - 0 = \frac{1}{2}T(1 + 1/N)$$

which are exactly those observed in the time-series analysis.

Similarly, $W(z)$ can be compared to $F(s)$, the continuum model. If its magnitude and phase closely approximate $F(s)$ within the I/O passband, the discrete model exhibits negligible I/O delay. However, to create consistent discrete code, the design objective at the subcycle level is to create a discrete realization such that

$$z^{-1/N} G_i(z)_N = z^{-1/N} \mathcal{Z}\{H_i(s)F(s)\}_N \simeq F(s)$$

If this is accomplished, a well-coded model ensues, because multirate variables are properly advanced for use in succeeding cycles. Unfortunately, this is inconsistent with the elimination of I/O time delays, which is only achievable for single-pass simulations ($N = 1$) because $L(z)$ introduces a time delay for multirate models.

Well-coded models thus display an advance of T/N at the multirate level, as in the cited first-order data-hold case. Unfortunately, they also manifest an I/O time delay entirely caused by the multirate phenomenon. This delay is a consequence of the rate conversion factor $L(z)$ only, because the computer delay per subcycle ($z^{-1/N}$) is canceled by the time advance in $G(z)_N$ for each subcycle. In this case $L(z)$ produces the phase shift

$$\phi = -\frac{1}{2}\omega T(1 - 1/N)$$

or time delay ΔD_1 . Hence, for well-coded models, a time delay penalty is invariably associated with multilooping. The

delay converges to the value $(1/2)T$ for large N . Since this time delay is constant, it is more properly referred to as a transport delay. "Transport delay" is defined here as the time delay of a discrete realization, obtained from phase measurements relative to the desired continuum system. Well-coded models demonstrate an invariant transport delay over a reasonable range of frequencies.

As has been shown, models not in this class are degraded by both $L(z)$ and the subcycle delay $z^{-1/N}$. For example, formulations that ignore the required advance in $G(z)_N$ show decreasing time delay for increasing N (with the same limit). Hence, if $G(z)_N$ has only the phase characteristics of $F(s)$ and does not include the cyclic advance, the time delay obtained from $z^{-1/N}L(z)$ is ΔD_2 .

Models displaying an effective triangular data hold have this delay if an advance, which is needed for the proper synchronization of multirate variables, is not present somewhere in the computations. The Ames simulation system, for instance, has such an advance in its kinematic model.

Models with an effective zero-order data hold accrue a delay of $(1/2)T$, independent of N . Note that for large N this is the general limit.

If a computationally unstable simulation model can be stabilized by use of alternate algorithms, thereby avoiding aliasing and the multirate delay penalty, the resultant model will probably be equivalent to, or even better than, that obtained using multiple loops, unless "properly designed multirate digital compensators" are implemented (Ref. 3, p. 353). Hence, the delay caused by these more stable algorithms may sometimes be tolerable in simulating networks of transfer functions.

If multirate processes are not used ($N = 1$), the discrete input-output relationship reduces to

$$\frac{r(z)}{p(z)} = z^{-1} \mathcal{Z}\{H(s)F(s)\} = z^{-1} G(z)$$

which again shows that the model should be predictive to compensate for the one cycle of delay. It also shows that delays or advances should be represented explicitly in the ESM; they should not be implied within transfer functions. This means that $G(z)$ should represent a concurrent transfer function, such that its output representation explicitly shows its advancing functionality, if any, with respect to the input time.

Concurrent Transfer Functions

Concurrent transfer functions relate the output to the time point of the input; any time shift is shown explicitly in the output representation. An example is given here to avoid the confusion that sometimes arises in reducing difference equations to z -transform notation.

Confusion can be avoided in convoluting difference equations to z -transform notation by taking the most recent

input as having the subscript k and letting the output receive its appropriately shifted index. Then variables with the subscript k are assigned the power $z^0 = 1$ in the z -transform representation. For example, the Euler integration algorithm can be represented by

$$v_{k+1} = v_k + T w_k$$

where the temporal advance is obvious from the subscripts and can be shown explicitly in the z -transform output representation,

$$\frac{zv(z)}{w(z)} = \frac{T}{1 - z^{-1}}$$

In this preferred form the error relative to perfect integration ($1/s$) is given by the ratio of performance to desired performance,

$$\begin{aligned} \frac{1}{1/s} \left[\frac{zv(z)}{w(z)} \right] &= \frac{j\omega T}{1 - z^{-1}} \\ &= \frac{\omega T [\sin \omega T + j(1 - \cos \omega T)]}{2(1 - \cos \omega T)} \end{aligned}$$

where the phase error is $\phi_k = (1/2)\omega T$. Since time delay has been defined as the negative of the phase error divided by the frequency, this expression produces a time delay of $-(1/2)T$, or rather, a time advance of $(1/2)T$ with respect to the input point. When the computer delay T is explicitly included, the output at the end of the cycle shows a net delay of $(1/2)T$ with respect to the expected output at the end of the cycle.

The representation of a simulation model using concurrent z -transforms is an important feature of the ESM, in which delays due to procedural code (computer cycles) are shown explicitly. An illustration is provided in the next section, of a model implemented using the Ames techniques.

A Simulation

Consider the second-order transfer function

$$F(s) = \frac{K}{s^2 + Ls + M}$$

where $K = 355.3$, $L = 26.39$, and $M = 355.3$. This system has an undamped natural frequency of 6π (3 Hz) and a damping ratio of 0.7. Because of the constant parameters and the lack of nonlinear elements in this system, the transfer function could be implemented using discrete transfer function techniques. However, for more generality, the parameters are not here considered as constants, and the addition of nonlinear elements (in simulation) is not precluded. A model containing a nonlinearity usually requires that the integration processes be separated.

The second-order system may be illustrated in a discrete realization by explicitly including the computer delays

caused by feedback paths and the two integrations $I_A(z)$ and $I_T(z)$, as shown in Fig. 5. The multirate z -transform $G(z)_N$ is thus

$$\frac{z^{1/N} u(z)}{p(z)} = \frac{KI_A(z)_N I_T(z)_N}{1 + z^{-1/N} I_A(z)_N [L + MI_T(z)_N]}$$

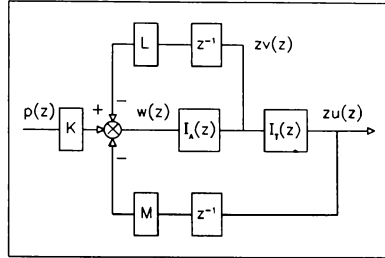


Fig. 5. Second-order model

In this form the predictive responsibility of the integration $I_A(z)$ is well illustrated. In the flight simulation facility at Ames this particular integration is handled by the Adams-Bashforth algorithm, which may be derived from $H_1(s)/s$, and it may be represented by the difference equation

$$v_{k+1} = v_k + \frac{1}{2}T(3w_k - w_{k-1})$$

At the multirate level this has the z -transform (again avoiding confusion)

$$I_A(z)_N = \frac{z^{1/N} v(z)}{w(z)} = \frac{(1/2)\tau(3 - z^{-1/N})}{1 - z^{-1/N}}$$

which manifests considerable phase lead, as required. The second integration algorithm at Ames is derived from the triangular data hold, and it is usually called the trapezoidal algorithm. Since this algorithm is not a predictor, it is naturally in concurrent form:

$$u_k = u_{k-1} + \frac{1}{2}T(v_k + v_{k-1})$$

The z -transform of the trapezoidal algorithm $I_T(z)$ at the multirate level is given by

$$I_T(z)_N = \frac{u(z)}{v(z)} = \frac{(1/2)\tau(1 + z^{-1/N})}{1 - z^{-1/N}}$$

which does not produce phase error with respect to perfect integration. The combination of these two algorithms has two important features pertinent to the discrete realization of vehicle kinematics in the Ames simulation environment. The acceleration-to-velocity integration process absorbs the burden of advancing the phase, approximating a temporal

advance of one computer cycle. The velocity-to-position integration algorithm then maintains the correct phase relationship (90° shift), ensuring that consistent velocity and position elements are available. Concomitantly, this scheme relieves programmers of the burden of time-shifting the outputs of various subsystem modules; hence, the triangular hold may be widely used, as recommended.

From the preceding discussion, we see that the total computer delay T is absorbed at the subcycle level by the term $z^{-1/N}$ in the ESM. This is shown in Fig. 3, where the computer delay is given per subcycle. $G(z)_N$ must then manifest the phase lead that is representative of one subcycle, $\tau = T/N$, in addition to the deliberate model phase characteristics of $F(s)$.

Example

As an example of the accuracy of the ESM, we consider the case in which the entire simulation model consists of just the Adams' integration algorithm, $G(z) = I_A(z)$, and we create the error with respect to perfect integration. Consulting Fig. 4, we see that we must include both the subcycle time delay and the rate conversion factor $L(z)$ in a multirate model. Using mixed notation, this produces the relative error

$$E(z) = W(z)/(1/s) = j\omega G(z)_N z^{-1/N} L(z) \\ = \frac{j\omega T(3 - z^{-1/N})z^{-1/N}(1 - z^{-1})}{2N^2(1 - z^{-1/N})^2}$$

from which the phase error is calculated:

$$\phi_e = \tan^{-1}\{\sin(\omega T/N)/[3 - \cos(\omega T/N)]\} - \frac{1}{2}\omega T$$

For determining transport delay, only the low-frequency region is of interest; this is easily extracted from the preceding equation by use of the small-angle ($\omega T/N$) assumption. This produces ΔD_1 . Since this is the same as the delay previously produced from examining just the rate conversion function $L(z)$, we here have a well-coded model (manifesting the multirate delay for $N \neq 1$).

The exact phase error can also be determined in this case. This requires an alternate derivation, because for comparison purposes we must avoid the ESM. Consider the sum of a geometric progression,

$$\sum_{n=0}^{N-1} z^{-n/N} = \frac{1 - z^{-1}}{1 - z^{-1/N}}$$

Substitute this into the expression for $I_A(z)_N$. The algorithm then takes the form

$$I_A(z)_N = \frac{1}{2}\tau \left[1 + 2(1 - z^{-1})^{-1} \sum_{n=0}^{N-1} z^{-n/N} \right]$$

For constant inputs for each of N consecutive subcycles, the summation in the above equation collapses to the sum of N constants:

$$\sum_{n=0}^{N-1} z^{-n/N} \Rightarrow N \quad (\text{with constant inputs})$$

Use of this substitution eliminates the subcycles from the expression for $I_A(z)_N$, as well as the requirement for rate conversion. That is, using lower brackets ("floor" operator) to denote the least integer operation, the sequence

$$v_{k/N} = v_{(k-1)/N} + \frac{1}{2}\tau\{3p_{[k/N]} - p_{[(k-1)/N]}\}$$

and the sequence

$$v_K = v_{K-1} + \frac{1}{2}\tau\{(1 + 2N)p_K - p_{K-1}\}$$

deliver the same answers for each $k/N = K$. The entire simulation model, including the total computer delay z^{-1} , then becomes

$$W'(z) = I_A(z)_N z^{-1} = \frac{(1/2)\tau(1 + 2N - z^{-1})}{1 - z^{-1}} z^{-1}$$

This is an exact expression for the response of the discrete system. Its error with respect to perfect integration is given by the relative error

$$E'(z) = W'(z)/(1/s) = j\omega W'(z)$$

which produces a phase error of

$$\phi'_e = -\tan^{-1} \left[\frac{\tan[(1/2)\omega T][N - \cos(\omega T)]}{1 + N - \cos(\omega T)} \right]$$

Extracting just the small-angle (ωT) performance from this expression also gives the delay ΔD_1 .

By comparing z -transforms we can get a feel for the accuracy of the approximations. Forming either $E(z)/E'(z)$ or $W(z)/W'(z)$ and extracting the relative time error gives the results shown in Fig. 6 (the error is zero for $N = 1$).

Figure 6 demonstrates that for frequencies much less than the 1/0 Nyquist limit, the differences are negligible. Since the "primed" expressions are an exact model of the system, the time delay computed from the ESM in this case displays negligible error.

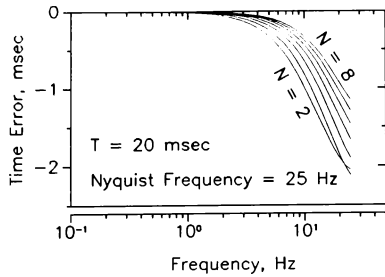


Fig. 6. Adams' ESM error.

The ESM should accurately predict the design phase characteristics for channels represented in z -transform notation. It has been our experience that if this is not the case, then either the model is not well coded, or the cycle time is not compatible with the frequency content of the system.

For more complicated simulation models the analysis of error using the ESM is best handled by time-series analysis, because the algebraic equations become formidable.

Additional Components

The consequences of assumptions (such as linearity) in the ESM are well beyond the scope of this paper, and the reader is invited to explore the limitations of z -transforms in the cited references. However, the benefits of the ESM are indeed great, and for the simulation of flight vehicles the approximations are of little consequence. This is because the spectra of aircraft responses are not competitive with typical I/O Nyquist frequencies in simulation.

The most important feature of the ESM is its multiplicative form for individual components,

$$W(z) = \Pi W_i(z)$$

which means that the individual phases add, as do the effective time delays,

$$D = \Sigma d_i = -\Sigma \phi_i / \omega$$

Each component of the simulation system may thus be considered independently.

We now turn our attention to components other than the simulation model itself and to a demonstration of their inclusion into the multiplicative ESM. An overview of the components included in this model is given by the architecture of Fig. 7, where the time delays ρ_i are caused by the components $W_i(z)$.

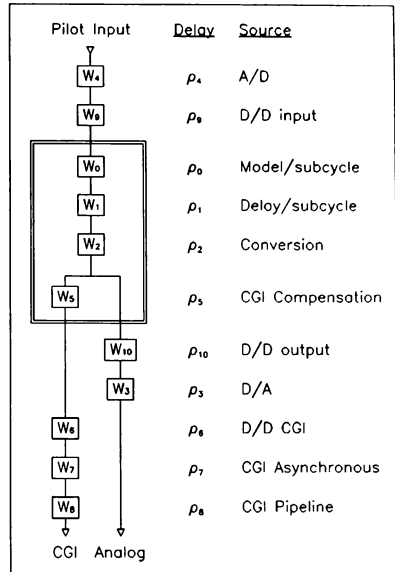


Fig. 7. Equivalent system model.

The following delay sources, listed here in the notation of Fig. 7, have already been discussed: the simulation model per subcycle, $W_0(z) = G(z)_N$, which, when divided by the baseline function $F(s)$, is assumed to have a relatively constant time delay ρ_0 for low frequencies; the transport delay function per subcycle, $W_1(z) = z^{-1/N}$, which produces a time delay $\rho_1 = T/N$ per subcycle; and the conversion function, $W_2(z) = L(z)$, which produces the multirate delay penalty $\rho_2 = (1/2)T(1 - 1/N)$.

Analog Output

The digital-to-analog signal (D/A) is given to a good approximation by

$$r_A(z) = z^{-(1/2)} r(z)$$

This is demonstrated as follows. The Newton-Gregory extrapolation formula for creating representations of data holds is developed in Ref. 2, pp. 31-34. By including just the first term of the formula, the zero-order data hold discussed in the section "The Discrete Multirate Model" is produced, as developed on pp. 34-36 of Ref. 2 and pp. 42-46 of Ref. 3. This may also be written

$$H_0(j\omega) = T h_0(\omega T) e^{(1/2)j\omega T}$$

where the gain factor,

$$h_0(\omega T) = \frac{\sin[(1/2)\omega T]}{(1/2)\omega T}$$

remains close to unity for small ωT products, and is often ignored in sampled-data systems. As developed in Ref. 2, pp. 123-124, when this element is combined with a plant with the transfer function $K(j\omega)$, the overall loop pulse transfer function is a complex expression approximated by

$$HK^*(j\omega) \simeq h_0(\omega T)e^{-(1/2)j\omega T}K(j\omega)$$

when conditions are such that $K(j\omega)$ is low pass relative to the Nyquist criterion. Notice that the factor T disappears. Furthermore, "for low frequencies, $[\sin(1/2)\omega T]/[(1/2)\omega T]$ is approximately unity, so that finally

$$HK^*(j\omega) \simeq e^{-(1/2)j\omega T}K(j\omega)$$

In words, the loop transfer function is approximately the equivalent of replacing the sample and hold operation by a pure time delay of half a sampling period" (Ref. 2, p. 124). Hence, the D/A conversion is

$$W_3(z) = z^{-(1/2)T}$$

resulting in the time delay $\rho_3 = (1/2)T$.

Conversion Delay

An analog-to-digital delay function is included in Fig. 7 as $W_4(z)$, producing the time delay ρ_4 . Although A/D dynamics produce a delay too small to be of concern here, low-pass prefilters are sometimes used to suppress electrical noise in a facility. Although this noise is generally of high frequency, it aliases into the Nyquist bandwidth during the sampling process. Since low-pass filters usually produce time delays in the low-frequency bandwidth, their phase characteristics must be measured and included in the ESM as the time delay ρ_4 .

At Ames Research Center the techniques of common mode rejection have succeeded in reducing laboratory noise. For example, the simulation complex that drives the Vertical Motion Simulator (VMS) does not require A/D prefilters. Hence, in a clean environment such as this, $\rho_4 = 0$.

The values of ρ_6 , ρ_9 , and ρ_{10} are all digital-to-digital (D/D) delays caused by discrete processors. Each of these components contribute approximately 2 msec of delay in the Ames simulation facility.

CGI Delay

A CGI system usually has projection logic within its first pipeline processor, which accounts for asynchronous data arrival from the simulation computer. The data are usually received via an interrupt into the first processor while the processor continues to work on previously acquired data.

The difference between the time of use and time of arrival is used for extrapolation and effectively cancels the asynchronous delay. This delay, given by ρ_7 , is therefore approximately zero. If the projection logic were disabled, the asynchronous delay would be approximately one-half the simulation computer's cycle time, $T/2$.

The CGI pipeline delay function, however, is significant. It is given by

$$W_8(z) = z^{-\rho_8/T}$$

The CGI pipeline delay is discussed in Ref. 8. It is generally on the order of 100 msec.

CGI Compensation

The CGI compensation scheme used at Ames is discussed in Ref. 8. It compensates for the pipeline delay ρ_8 by producing a phase lead, equivalent for small frequencies to a time lead ρ_5 (negative valued). From the reference, we see that the compensation algorithm uses the velocity history at the I/O frame rate. The algorithm may be expressed

$$\frac{u_c(z)}{v(z)} = \frac{u(z)}{v(z)} + \sum_{i=1}^N b_i z^{1-i}$$

where N is an odd integer, reported in the literature for the case of $N = 3$ (Ref. 8). The b_i coefficients are functions of the cycle time T , the projection interval P , the pipeline delay ρ_8 , and a tuned frequency ω_f , nominally set to about 3 Hz (see Ref. 8, p. 7). Using these parameters, CGI pipeline delay essentially vanishes in the pilot's bandwidth, as discussed in the reference, i.e., $\rho_5 + \rho_8 \approx 0$.

Consulting Fig. 7, we can summarize the ESM as follows: (1) The model W_0 contains the desired phase relationships plus one subcycle of time advance. (2) The delay-per-cycle penalty ρ_1 is one subcycle of delay. If the product $W_0 W_1$ constitutes a well-coded model, the delay caused by the discrete realization process vanishes. (3) The D/A penalty ρ_3 is one-half the mainframe cycle time. (4) The A/D penalty ρ_4 is negligible when a prefilter is not required or a good prefilter is used. (5) The CGI compensation algorithm W_5 cancels the pipeline delay ρ_8 for frequencies in the pilot's bandwidth. (6) Digital-to-digital delays are approximately 4 msec; these occur as $\rho_9 + \rho_6$ for the A/D through CGI path, and as $\rho_9 + \rho_{10}$ for the A/D through D/A path. (7) The asynchronous penalty ρ_7 vanishes because of projection logic. (8) The multirate penalty ρ_2 vanishes if multiple loops are not used.

In the Ames flight simulation facility, when a multirate model is not implemented, delay from the pilot to the CGI display is approximately 4 msec and delay from the pilot to the analog outputs of simulations is approximately $4 + T/2$ msec, or about 15 msec. These numbers are considerably less than the 150-msec range of values easily obtained from component-level experiments.

Conclusions

An equivalent system model may be created as the product of individual simulation components expressed as z -transforms,

$$W(z) = \Pi W_i(z)$$

Because of this relationship, the individual phases and component time delays are additive. This permits the independent investigation of various delay sources in flight simulation.

The time-delay penalty for well-coded multirate models (N loops per I/O cycle) is

$$D = \frac{1}{2}T(1 - 1/N)$$

Although procedures for handling multiple loops have been established, the use of a multirate model is not encouraged because of the I/O time-delay penalty and because of problems with aliasing. If alternate solutions are not available for attaining computational stability, then a multirate model should be accompanied by filters that attenuate spectra at the expanded rate.

Simulation input-output relationships that can be expressed in z -transform notation can be included in the equivalent systems model using mathematical relationships described in this paper. Alternately, time-domain software is capable of measuring delays, which may then be used in the equivalent systems model for quality control. A determination may then be made as to whether the model is well coded.

Computer algorithms, notably the predictive integration algorithm and the CGI compensation algorithm, are subsets of the equivalent systems model, as well as subsets of the

flight simulation system at Ames. They dramatically reduce total system delays during flight simulation from the delays observed during component-level experiments.

References

- ¹Paulk, Clyde Jr., Astill, David L. and Donley, Shawn T., Simulation and Evaluation of the SH-2F Helicopter in a Shipboard Environment Using the Interchangeable Cab System, NASA TM-84387, August 1983.
- ²Ragazzini, John R. and Franklin, Gene F., Sampled-Data Control Systems, McGraw-Hill Book Co., New York, 1958.
- ³Kuo, Benjamin C., Analysis and Synthesis of Sampled-Data Control Systems, Prentice-Hall, Englewood Cliffs, NJ, 1963.
- ⁴McFarland, R. E., The N/Rev Phenomenon in Simulating a Blade-Element Rotor System, NASA TM-84344, March 1983.
- ⁵McFarland, R. E., Quiet Mode for Nonlinear Rotor Models, NASA TM-102239, April 1990.
- ⁶Crochiere, Ronald E., and Rabiner, Lawrence R., Interpolation and Decimation of Digital Signals – A Tutorial Review, Proc. of the IEEE, Vol. 69, No. 3, March 1981.
- ⁷McFarland, R. E., and Rochkind, A. B., On Optimizing Computations for Transition Matrices, IEEE Trans. Auto. Control, Vol. AC-23, No. 3, June 1978.
- ⁸McFarland, R. E., Transport Delay Compensation for Computer-Generated Imagery Systems, NASA TM-100084, January 1988.

VISUAL ADVERSARY LOGIC FOR CLOSE-IN AIR COMBAT SIMULATION

Bruce Montag
 Southwest Research Institute
 San Antonio, Texas

Abstract

Traditional air combat adversary maneuvering logic models commonly used to control computer generated threats in flight simulators are limited in the provision of cues required for effective close in combat tactics training. After the merge, pilots rely almost exclusively on over the shoulder direct viewing of the adversary aircraft for tactical maneuver decision making. Experienced pilots can quickly determine adversary intent, capability, and relative advantage by glimpsing and visually tracking the adversary in relation to his own aircraft during dogfight engagements.

Current 5 degree of freedom (DOF) adversary performance models used in weapon system trainers and engineering simulators provide limited engagement logic for close in combat simulation. When within visual range, these engagement models typically drive the adversary on a pure pursuit course for offensive maneuvering. This type of guidance law results in adversary flight paths that are not totally indicative of operational scenarios and do not communicate key tactical parameters to the engaged simulator pilot.

The 5 DOF performance model can be enhanced significantly by incorporating visual adversary logic in the threat aircraft simulation. Realistic flight paths and engagement events can be produced by improving the simulation of close-in-combat fire and flight control characteristics of the adversary aircraft. The fire control mode and flight tactic selected by the threat pilot determine the type of guidance law that he will fly his aircraft by, resulting in a flight path that clearly indicates his tactical intent and ability. By correlating realistic out-the-canopy visual adversary flight path characteristics with cockpit sensor indications, the simulator pilot can practice and train in the art of close-in visual range combat.

Nomenclature

a_i	Airframe acceleration
θ	Euler pitch angle
ϕ	Euler roll angle

Ψ	Euler heading angle
V	Aircraft true airspeed
N	Load Factor
P	Body roll rate
ρ	Air density
K_i	Aircraft specific aerodynamic factors
H	Altitude
E_s	Specific energy
P_s	Excess specific power
C_L	Lift coefficient
C_D	Drag coefficient
S	Effective airframe surface area
α	Angle of attack
τ, γ	Flight path angles
β	Ballistic drop angle
Z_{drop}	Ballistic drop distance
Δt	Iteration interval
TOF	Time of flight
V_{avg}	Average projectile velocity
θ_{lead}	Lead angle elevation error
ϕ_{azerr}	Bank error due to lead angle azimuth error

Introduction

This paper presents several simple yet innovative visual adversary enhancements to the traditional five degree of freedom performance model that is commonly used to fly adversary aircraft in weapon system trainers and engineering simulators. The enhancements add variable lift to adversary flight characteristics and provide a modular command structure for integrating fire and flight control logic into the aircraft performance model. Fire control modes considered include aerial gunnery modes such as lead computing optical sight (LCOS) gunnery mode and snapshot gunnery mode. A key feature of the simulated aerial gunnery modes is an innovative guidance law that accurately accounts for ballistic trajectory attributes with minimum computation requirements.

Close-In Combat Visual Cues

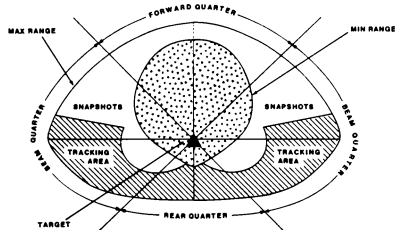
Close-in tactical 1-vs-1 maneuvering decisions are driven almost exclusively by visual information derived at close range. This visual information consists of the pilot's perception of how the other aircraft is moving relative to his own, and is highly event driven. Aircraft state changes, gun barrel muzzle flashes, rocket motor smoke plumes, afterburner glow, and wing surface movements are all visual combat event indicators that must be evaluated and responded to with split second maneuver decisions.

Air-to-Air Gunnery Tactics

The gun is the key weapon for "knife range" fighter combat. Knowledge of air-to-air gunnery tactics and associated offensive and defensive maneuvers is critical to combat success once an engagement has devolved into a close range dogfight.

Offensive Tactics

The offensive gunnery objective is to fly the aircraft to achieve a high probability of kill firing opportunity. A firing opportunity occurs when the target aircraft and the shooter's gun trajectory occupy the same piece of airspace at the same point in time. This feat can be accomplished through a variety of tactics, and results in two basic gunnery engagement situations—the high deflection snap shot, and the lead collision tracking shot. The effective envelopes for each type of situation is shown below.¹

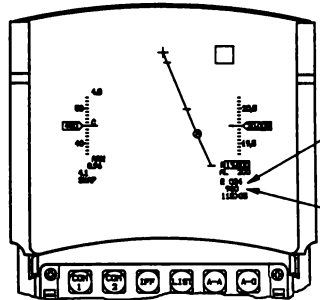


Gun Engagement Envelope

Snapshot

The snapshot method derives its name from the fact that in snapshot mode, the target and gun trajectory are aligned for only an instant or two, and thus only a "snapshot" firing opportunity is presented to the shooter.

Snapshot gunsights are typically implemented via a historical trajectory type display. The trajectory is displayed as a continuously computed impact line (CCIL) that shows the historical location of simulated shells fired one time of flight (TOF) previously.¹



Snapshot Mode Gunsight

The pilot's task for snapshot engagements is to position his aircraft such that the target flies through the CCIL pipper within effective gun range. Achieving this condition requires the shooter to mentally project the targets flight path and steer in an appropriate amount of lead angle to place the target under the CCIL pipper. Varying amounts of load factor are usually required to do this, and pipper convergence will occur only briefly due to the relative motion between the two aircraft.¹

Lead Collision Tracking

The other type of gun engagement is the tracking shot, which is performed through the use of a LCOS type sight. LCOS sights use a disturbed reticle type display to indicate the bullet impact point.



Defensive Tactics

Break Turn

Turn Reversal

Jink

Extension

Visual Adversary Simulation

An alternative approach is to add visual adversary logic to the simple 5 DOF performance model that is commonly used in training simulators. Simulators that employ combat simulation programs such as TAC Brawler and Adaptive Maneuvering Logic (AML) could also benefit from the incorporation of visual logic for close-in tactics training.^{2,3}

The 5 DOF performance model is the most widely used model for implementing airborne adversaries.⁴ This model utilizes point mass aircraft characteristics to determine the acceleration, velocity, position, and attitude of the adversary aircraft. Forces acting on the aircraft's center of mass are limited to aggregate values of lift, drag, thrust, and weight. Since sideforce (i.e., rudder) modeling is not included, the coordinated bank-to-turn method is used to control aircraft heading. Linear airframe accelerations in local level coordinates are then computed as shown in equations 1 through 3.

$$a_x = \left(\frac{\text{Thrust} - \text{Drag}}{\text{mass}} \right) - \left(\frac{\text{Weight} \times \sin \theta}{\text{mass}} \right) \quad (1)$$

$$a_y = \left(\frac{\text{Lift} \times \sin \theta}{\text{mass}} \right) \quad (2)$$

$$a_z = \left(\frac{\text{Lift} \times \cos \theta}{\text{mass}} \right) - \left(\frac{\text{Weight} \times \cos \theta}{\text{mass}} \right) \quad (3)$$

Airframe directional velocity and position within the gaming area is then computed by performing successive integrations on the acceleration vector and resulting velocity vector. Altitude rates are determined as a function of an arbitrary roll rate and excess lift force, as shown in equations 4 through 6.

$$\dot{\theta} = \frac{a_z \cos \Phi}{V} \quad (4)$$

$$\dot{\Psi} = \frac{a_y}{V \cos \theta} \quad (5)$$

$$\dot{\Phi} = P + \Psi \sin \theta \quad (6)$$

Aircraft orientation angles in local level coordinates are then computed by integrating the attitude rates.

Controlling the adversary flight path with the 5 DOF performance model is simply a matter of selecting the desired bank angle, load factor, and true airspeed needed to place the aircraft on course. Aircraft heading rate is a function of bank angle, which is controlled by roll rate, and excess lift force, which is a function of load factor. Load factor is the primary determinant of lift and drag force, as shown in equations 7 and 8.

$$\text{Lift} = N \times \text{Weight} \quad (7)$$

$$\text{Drag} = \frac{1}{2} \rho V^2 K |N| \quad (8)$$

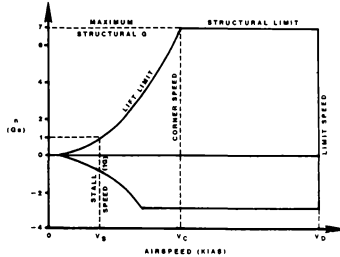
Aircraft altitude rate is a function of true airspeed and pitch angle, which is controlled through the lift to weight ratio. Flight path control is typically accomplished through simple relationships between desired heading/altitude and load factor/roll rate.

Performance Model Limitations

The main limiting feature of the 5 DOF performance model is that energy characteristics are too simplistic for visual

dogfighting. The model is entirely adequate for beyond visual range (BVR) engagements, but it is not suitable for dynamic close-in combat situations. Although absolute g limiting can be imposed, adversaries are stall proof under the performance model and unrealistic turn rates are possible.

A 6 DOF stability model automatically accounts for energy characteristics through the variability of aerodynamic force coefficients. In reality, the available load factor and turn rate performance are functions of airspeed, altitude, and aircraft configuration.¹



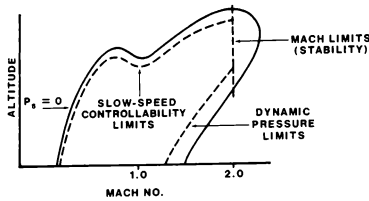
Aircraft Load Factor Envelope

Energy characteristics modeling is critical to providing the proper cues necessary for visual range dogfighting. The visual adversary model must provide for energy transfer between kinetic and potential energy, and the change in energy state due to the application of power. The specific energy state of the aircraft is a function of potential and kinetic energy as shown in equation 9. The rate of change in energy state is known as specific excess power and is shown in equation 10.

$$E_s (ft) = H + \left(\frac{V^2}{2g} \right) \quad (9)$$

$$P_s (ft/sec) = \left(\frac{\text{Thrust} - \text{Drag}}{\text{Weight}} \right) V \quad (10)$$

Diagrams describing the performance envelope of aircraft are created by mapping regions of constant P_s against altitude and airspeed.¹



Aircraft Performance Envelope

Visual adversary models must account for energy performance to produce realistic turn rate, pitch rate and climb rate characteristics. For example, when a wind up turn is performed, either altitude or airspeed must be reduced to maintain the turn rate.

Visual Adversary Logic Enhancements

Several simple enhancements to the 5 DOF performance model are available that increase the utility of the model for visual range tactical engagement simulation. These enhancements include incorporation of variable lift characteristics into the flight dynamics equations, the addition of a tactical command level autopilot for modeling integrated adversary fire and flight control modes, and the incorporation of tactical guidance logic.

Variable Lift

Variable lift effects can be modeled by associating lift force directly with angle of attack and dynamic pressure. Simple polynomials permit lift and drag coefficient behavior to be modeled as a function of angle of attack as shown in equations 11 and 12. The resulting lift and drag forces acting on the airframe are determined in equations 13 and 14.

$$C_L = C_{L_0} + K_1 \alpha - K_2 \alpha^3 \quad (11)$$

$$C_D = C_{D_0} + K_3 C_L^2 \quad (12)$$

$$Lift = C_L QS \quad ; \quad Q = \frac{1}{2} \rho V^2 \quad (13)$$

$$Drag = C_D QS \quad (14)$$

Angle-of-attack can be computed as a function of commanded load factor, aircraft weight, and dynamic pressure.

$$\alpha = \left(\frac{N \times Weight}{QS} \right) K_4 \quad (15)$$

Aircraft attitude is then computed by accounting for angle of attack in the local level coordinate frame.

$$\Psi = \tau + \alpha \sin \Phi \quad (16)$$

$$\Theta = \gamma + \alpha \cos \Phi \quad (17)$$

Tactical Command Autopilot

A tactical command autopilot can also be configured to produce virtually any flight path using the enhanced 5 DOF performance model. The autopilot consists of a pitch channel and a roll channel that can be driven by variable stick inputs (i.e. fore/aft and left/right stick deflections or force inputs) or by command values. The autopilot is configured such that adversary flight can be controlled either automatically or by a joystick. Stick inputs always override command values.

Pitch Channel

Pitch channel command values consist of either target altitude or target pitch angle. Altitude commands are converted into target pitch angles by the pitch channel of the autopilot. The difference between the current and desired pitch angles is used to calculate a proportional pitch rate command. A commanded load factor is then determined from the desired pitch rate and aircraft velocity.

$$N_c = \left(\frac{Q_c V}{1845} \right) \quad (18)$$

To achieve realistic aircraft performance, the command load factor is run through a low pass filter, and is limited based on the maneuvering characteristics of the adversary.

Use of the low pass filter results in smooth angle of attack rotation performance, which is critical for within-visual-range maneuvering. The variable lift angle of attack feature allows for over rotation (i.e. stalling) and provides for realistic energy transfers in turns.

Roll Channel

Roll channel command values are either target heading or target bank angle. A body roll rate command is generated based on the difference between current and desired bank angle. Since all adversary turns are assumed to be coordinated, target heading inputs are converted into bank angle inputs, subject to load factor limitations. Unloaded body rolls can also be performed for roll-to-bank angle maneuvers. Output roll rate

is limited as a function of airspeed and maneuvering characteristics of the adversary.

Tactical Maneuvers

The tactical command autopilot permits complex maneuvers to be performed by dividing the maneuver into flight phases, with each flight phase described by a set of autopilot input commands. For example, a 135 degree slice back maneuver might be constructed from a pitch up phase, an unloaded roll over phase, and a pull through phase. This approach allows tactics libraries to be developed that are capable of describing virtually any maneuver.

By incorporating tactical geometry into the flight path command process, integrated fire and flight control guidance commands can be generated in real-time. This capability allows the enhanced 5 DOF performance model to simulate various fire control guidance methods such as Lead Computing Optical Sight (LCOS) and snapshot gunnery modes.

Aerial Gunnery Guidance

Visual adversary use of automated gunnery modes can be effectively simulated using a 5 DOF performance model with visual adversary logic improvements. Guidance commands for orienting the adversary to achieve a guns solution is achieved by modeling the gun line-of-fire trajectory. The key to gun trajectory modeling is the ability to calculate the amount of ballistic drop away from the gun firing line. For aerial gunnery, the drop distance was found to be a function of time of flight and bullet distance traveled.

$$Z_{drop} = \left(\frac{dZ/dX}{dt} \right) \times \left(\frac{TOF}{\Delta t} \right) \times distance \quad (19)$$

The ratio between drop distance and distance traveled was found to be relatively constant ($dZ/dX/dt = .0007$) for a 20mm shell for the first two seconds of bullet flight. After two seconds, the shell generally has lost too much energy to be considered effective.

For guidance purposes, the ballistic drop distance can be converted to a drop angle off of the gun line-of-sight.

$$\beta = \arcsin \left(\frac{Z_{drop}}{range} \right) \quad (20)$$

LCOS gunnery mode is simulated by entering an integrated fire and flight control tracking loop when the adversary visually acquires the ownship aircraft. A quadratic prediction filter is used to estimate the future position of the ownship relative to

the adversary given the observed states of both aircraft. Adversary stick and throttle commands are then generated to aim the adversary towards the ownship's predicted position for a one bullet time of flight.

$$TOF = \frac{-range}{range \ rate - V_{eq}} \quad (21)$$

Stick commands are then generated to null the line of sight rate between the adversary and the projected ownship position, while throttle commands are adjusted to maintain a reasonable closing velocity on the ownship.

$$\Phi_{LCOS} = \Phi_{serv} + \arctan \left(\frac{\dot{\Delta z} \times V \cos \theta}{1845} \right) \quad (22)$$

$$\theta_{LCOS} = \theta_{serv} + \beta \quad (23)$$

Gun firing during LCOS mode operation is initiated when the range between adversary and ownship is within the gun envelope and the LCOS azimuth and elevation steering errors are brought to within fire control limits.

With the addition of an out-of-plane set up maneuver, the basic gunnery guidance scheme can also be used to effectively model snapshot attacks. The main difference being that snapshot mode typically includes greater lead angle positioning and wider fire control limits.

A total close-in visual environment can be created by adding additional visual adversary logic to determine afterburner status, speedbrake position, muzzle flash, and gun trajectory flyout. The addition of these visual clues plus the close-in gun tactics and realistic maneuvering characteristics provide for a simple, yet effective visual adversary model for dome visual equipped simulators.

Conclusion

Incorporation of the suggested visual adversary logic enhancements in domed simulators would allow pilots to associate visual threat aircraft behavior with relative tactical advantage. For instance, pilots could learn to recognize and differentiate between specific close-in combat tactics such as LCOS or snapshot engagements, quickly assess the adversary's ability to achieve a fire control solution, and perform a corresponding defensive or offensive conversion maneuver.

With visual adversary logic enhancements, all the necessary visual cues are provided such that the ownship pilot can determine if the simulated adversary pilot is attempting to pull lead, is tracking in the groove, or is out-of-plane and setting up for a snapshot. Current adversary maneuvering logic models do not offer this level of dynamic interaction between pilot and simulated threat for close-in combat visual tactics training. The addition of visual adversary logic to the 5 DOF performance mode could significantly enhance the training capabilities of a simulator, without resorting to computationally expensive and data intensive 6 DOF stability models for interactive adversaries.

References

- 1) Fighter Combat--Tactics and Maneuvering, Robert L. Shaw, Naval Institute Press, 1986.
- 2) Interactive Computerized Air Combat Opponent, Walter Hankins, AGARD Flight Simulation Conference, 1975.
- 3) TAC Brawler Air Combat Simulation Analyst Manual, Decision Science Applications Report #668, October 1985.
- 4) Aerodynamic Math Modeling, R.T. Galloway, Flight Simulation Update, Binghamton New York, 1988.

USER DEFINABLE DOCTRINE FOR INTERACTIVE AIR TARGETS

R. Ure and D. N. Siksik
CAE Electronics Ltd.
St-Laurent, Quebec, Canada

ABSTRACT

The requirement for realistic tactical environments on training devices has grown for economic and political reasons. Also, the rapid growth in the capabilities of computer hardware allows the development of very sophisticated systems which can run in real time.

This paper describes an important element of the tactical environment, the interactive air target. CAE's interactive air target system is designed to provide intelligent air combat adversaries whose decision, response and performance capabilities are user definable via an expert system interface.

There are three main components to this system. The vehicle model, a knowledge base and an inference system. The vehicle performance and systems are user definable. The knowledge database is constrained within limits which ensure realistic simulations of a set of acquired and underlying rules. The acquired rules are definable by the operator off-line via an interactive rules editor. The underlying rules are basic to the structure and provide a framework for the operation of the inference system. The inference system will integrate the information provided by the air target sensors, active and passive, process the reduced data according to the acquired and underlying rules to ultimately arrive at a course of action.

The tools used to develop interactive air target performance and doctrine are described in this paper as well as the reusability and flexibility of the software generated by these tools.

INTRODUCTION

Many current real time tactical simulations are lacking in the key area of the simulation environment. More specifically, there exists a requirement for realistic elements within the environment which fill the role of a challenging adversary or comparative ally, in training, or of an experimental platform, in system development.

The Interactive Air Target (IAT) is a principle component within CAE's Interactive Tactical Environment Management System (ITEMS)¹. Whether it is used as part of an ITEMS generated scenario or on its own, as a stand-alone system, the IAT provides a flexible and intelligent entity to the simulation environment. The IAT concept is based on its ability to adopt a wide variety of characteristics, limited only by the amount of data defined within the IAT's Database Management System (DBMS).

To support the characteristics of aircraft and systems defined within the DBMS, the IAT makes use of control objects, such as A/C dynamics modelling, systems modelling, ballistics and scoring. Other simulation objects include navigation and position keeping.

IAT intelligence is achieved by employing an inference engine to decode and carry out rule-based doctrines (knowledge base) already defined in the DBMS. The engine uses simulation input (provided by the objects discussed above) to build parameters for decision making by the doctrine (see figure 1).

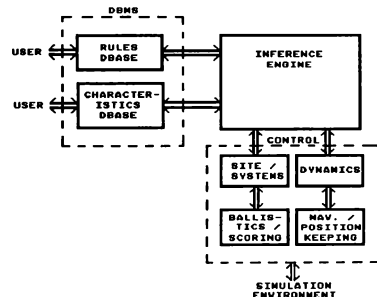


Figure 1: IAT System

The following will describe the IAT in terms of its building blocks: the DBMS, the control objects and the inference engine.

IAT CHARACTERISTICS (DBMS)

The Database Management System enables characteristics for IAT systems and doctrines to be applied to the IAT in a flexible manner. Furthermore, it provides a user-friendly interface for inserting and modifying data. The information within the DBMS is used by the control objects and inference engine within the simulation in order to achieve the effects designed by the user (see figure 2).

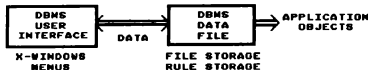


Figure 2: DBMS System

Definition of DBMS knowledge takes a "bottom-up" approach on two separate levels: library data or scenario data. The former is organized into volumes of very specific information which may then be used by higher level descriptions. For example: the radar mode library is referenced within the radar library which is, in turn, accessed by the IAT player library. The scenario design level, on the other hand, combines the highest level libraries (IAT player, doctrine) into the defined layout of a tactical scenario.

IAT PLAYER LIBRARY

The IAT player library holds many different versions of the IAT with respect to its physical characteristics and systems. Each composition occupies one record of DBMS data and includes the following specifications:

Platform:

The IAT may be specified to be either a fixed or rotary wing aircraft. A further description is implemented for use in picking the most appropriate visual model. Platform details such as dimensions, motion characteristics and laser reflectivity are also declared.

Fuel:

Fuel load and consumption are defined.

Zones:

For each IAT description, the platform surface may be broken down into zones made of four-sided polygons. Zones may be specified for: (i) the designation of equipment on the player (A system assigned to a given zone will be considered by the system controllers and ballistics to be at the centre of that zone.) and (ii) the assignment of vulnerability and manoeuvring degradation characteristics for the purpose of scoring (Each zone is assigned a vulnerability index which, when surpassed by a damage index inflicted via the scoring object, flags the destruction of that zone, any equipment located within it as well as the appropriate degradation in IAT manoeuvrability.).

Gun Systems:

Specifications include turreted or non-turreted options as well as the gun's physical turning limits and its position on board the platform. The gun itself is defined to great detail within the guns library and depends on even finer definitions within the rounds library.

Rocket and

Missile Systems:

Assignment of Rocket and Missiles depend on the allotment of weapon stations to the IAT platform via its zones. One weapon only may be placed at each zone and may have associated with it such characteristics as physical motion limits, quantities and firing intervals. The characteristics of each rocket or missile type is recorded in their respective libraries.

Sensors:

Both active and passive sensors may be assigned, including: warning receivers, FLIR, radar, IFF transponder, TV camera and crew visibility. As an exception, the radar library makes use of greater details defined in the radar modes library.

Countermeasures:

Jamming, chaff, flare and smoke systems are defined. Again, individual libraries are included for the definition of each countermeasure.

Communications:

Communications systems are defined for use with a simple communications scheme in an ITEMS generated scenario.

Player Signature:

Radar and thermal signatures are defined for each platform.

Dynamic Envelope:

Several characteristics which define the dynamic envelope of the IAT and which will be used by the dynamics control may be assigned.

Designation of characteristics to the dynamic envelope is restricted by a verification procedure using an identical flight model as does the dynamics control object. Only configurations which are verified to be stable in flight are accepted.

DOCTRINE LIBRARIES

As opposed to the IAT player library, the doctrine libraries provide for the intelligence of the IAT and constitute the knowledge base of the system.

A doctrine library holds several rule-based doctrines, each of which may be assigned to the IAT in order to achieve a desired behavior.

Each doctrine is built by a tactician who specifies rulesets in a hierarchical structure based on his identification of possible situations and appropriate reactions. A ruleset is a building block within the doctrine which simplifies the classification of the IAT's status. Each consists of a fixed number of if/then rules which are, in turn, built of condition, result and response fields. The result may call another ruleset, set a local variable or execute a response. Conditions may be based on local variables or like the responses may be chosen from lists of available parameters.

To facilitate the task of the tactician and to make the construction of sophisticated doctrines possible, many parameters are of a high level nature and require pre or post processing. As opposed to known quantities such as velocities and distances, these parameters can describe values not yet known but which may be derived: "the IAT opponent is within radar range" or "permission to fire missile".

The doctrine libraries include, first and foremost, a library of behavioral rules for general control of the IAT actions and a library of air combat rules to translate combat goals into manoeuvre goals.

The IAT is also provided with a set of rules for mission definition. Such rules basically use specific points defined either on a trajectory or via mission time in order to assign mission activity which may consist of one, or several, actions at each point.

Furthermore, libraries are provided for opponent selection rules, for trajectories as well as for manoeuvres which the IAT may use within its doctrine responses and which will be realized through its flight model.

The library data defined above is used in the scenario design level of the DBMS. At this point, the IAT is assigned a mission trajectory, a doctrine and a set of characteristics. The environmental and terrestrial conditions within the scenario are also assigned. If the IAT is part of an ITEMS generated scenario, there will exist many platforms or 'players' with which the IAT may interact and an opponent selection doctrine will be chosen. The other players would have their doctrines and characteristics assigned at this level as well.

The DBMS user-interface consists of X-windows generated menus for accessing and modifying the libraries as well as a Tactical Situation Display (TSD) for designing the scenario. The latter is a cartographical display of the scenario over which initial positions and trajectories as well as terrestrial and environmental data can be assigned.

IAT CONTROL

Once the characteristics of the IAT and its scenario are defined, via DBMS, a control structure must be imposed in various areas in order to manage the simulation. Of primary importance is the control imposed by the dynamics object which supplies the aircraft model and which relies heavily on navigation and position keeping. Also important is the system control object which enables the IAT systems to interact with their environment. In turn, ballistics and scoring, as well as communications, are relied upon to further support systems interaction. Each control object is briefly detailed below.

DYNAMICS

The dynamics control object provides the simulation with atmospheric, attitude and position data about the IAT according to the manoeuvre goal chosen by the doctrine and the dynamic envelope characteristics assigned through DBMS.

Within the dynamics, manoeuvre goals are interpreted and then implemented through such parameters as velocity, acceleration, altitude, turn rate, etc. which are input as demands to a control process

based on the flight model. Provided as output are the positions of control surfaces which include: elevators, ailerons, rudder and throttle. The latter are, in turn, input to the fully aerodynamic flight model which is customized by the dynamic envelope characteristics and which solves the equations of motion describing the flight path and attitude of the IAT. The results are attitude, acceleration and velocity information as well as position data, which is provided by the position keeping object. These data are fed back to the manoeuvres control thus implementing a large, closed-loop control system (see figure 3).

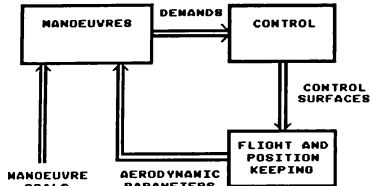


Figure 3: Dynamics System

NAVIGATION

The navigation control interfaces with the dynamics object and a terrain database to allow the IAT to adopt various navigational modes thus increasing the control of the doctrines and greatly improving the IAT's realism and capabilities. The navigational modes available are: direct, low level, contour and nap of the earth. In all cases, verification is made to ensure that physical requirements for a given mode are met by the systems provided for the IAT within DBMS.

POSITION KEEPING

Position keeping interacts with the navigation and dynamics controls in order to provide a constant update of the IAT position.

AIRCRAFT SYSTEMS

Control of systems interaction is provided through the use of simplified models of each system (eg: sensors, lasers, countermeasures). A complex model for missiles, rocket and guns systems is implemented by the ballistics control.

Sensor data is conveyed to the IAT via a situation buffer which records all direct sensor contacts as well as contact information provided through communications.

The systems object processes doctrine requests for radar and weapon control. It responds by enabling radar or firing weapons, if pre-set conditions for the actions (such as sufficient ammunition) are met. Requests for other systems control are similarly handled.

In order to provide effective control and reduce the burden of doctrine programming, systems responses are of a high level nature and provide intelligent control. For example, in the case of several radars assigned to the IAT, the more optimum system (as determined by its description within DBMS) will be automatically engaged on a 'radar on' request from the doctrine. The less qualified sets are left for backup or must be specifically requested. Similarly, intelligent handling for weapons exists. A permission to fire missiles command to the systems will co-ordinate:

- (i) the optimum missile;
 - (ii) identification of the opponent (as chosen by the IAT doctrine);
 - (iii) the opponent's image via the IAT sensors;
- in order to fire the missile at the most opportune time.

All requests to systems and all its actions are subject to verification with respect to original DBMS definitions in order to ensure the integrity of the design.

BALLISTICS

As does the dynamics control, the ballistics object provides the dynamics models for missiles, rockets and gun systems based on individual characteristics previously defined for each system and projectile within the DBMS. The model requires the projectile/missile origin vector, as given by systems, and determines the final path and destination taking such effects as wind and rotor downwash into account.

SCORING

Scoring uses the destination of the weapon (missile or projectile), as given by ballistics, and its damage producing level, as given by DBMS, in order to map damage onto the appropriate IAT zones, or

the zones of its opponents. The amount of damage sustained by any of the various systems is thereby determined. Upon the destruction of any system, the systems object is flagged allowing it to preclude the lost equipment from any further commands.

COMMUNICATIONS

Management of a simple communications scheme is available when the IAT is part of the ITEMS generated scenario. Players may be assigned to transmit various information according to specific requirements and using specific equipment and codes as defined in the DBMS. Information which may be accessed by the IAT in virtue of its on-board equipment is placed in the IAT's situation buffer.

INFERENCE ENGINE

Within the IAT system, the inference engine serves the role of interpreting and carrying out the rule-based doctrines provided by the DBMS.

The doctrine, having been input to the DBMS through a user-friendly menu/lists system, is converted, by a doctrine decoder, into a format which reflects the actual offsets of the doctrine parameters within the common database and is therefore legible by the interpreter.

With a useful rule format available as input, the interpreter can commence its rule evaluation. Conditions, which consist of up to two parameters and an operator are handled by acquiring the parameter values via a look-up table and applying them to the operator. If the rule fires, the result section is executed and consists of a choice of:

1. Interpreting another ruleset. This is equivalent to branching within the ruleset tree.
2. Ending a ruleset interpretation. Equivalent to ending a ruleset tree branch.
3. Setting a local variable. Permits variables used only locally within the doctrine to affect its path.
4. Execute current response only. The current response is executed immediately

and on its own. The doctrine interpretation ends.

5. Execute response. The response is executed as detailed below.

In the case of the latter, the response(s) is built by accessing the parameters to be modified (again via a look-up table) as well as their intended values and placing these in a response buffer under an appropriate response heading. Responses are classified as manoeuvre goals, RF jammer, IR jammer, chaff, flare, smoke, laser, radar and weapon systems. Upon completion of the doctrine interpretation, these latter are executed and doctrine evaluation is re-initiated.

OVERVIEW

With the inference engine in place, the IAT, itself, becomes a feedback system, as did many of its building blocks. Parameter values reflect the status of control objects (which depend on DBMS assigned characteristics); are used by the inference engine during rule interpretation of the DBMS doctrine; affect doctrine path execution (through conditions) and eventually (through doctrine responses) modify control object parameters, thus closing the loop.

A "bottom-up" approach having been taken within the DBMS, with the lowest layers having a large scope of detail and specification, the potential for a very flexible and therefore powerful air target simulation is available. Use of control objects with sufficiently complex models enables system interaction to be handled in a realistic manner.

ITEMS and IAT are currently under development. Applications include not only real time interactive air target simulation for training purposes but also simulation for systems development or for doctrine evaluation. Further potential exists for the simulation of other interactive vehicles, including submarines.

REFERENCES

- 1 Morris A., "The Creation of Complex Tactical Training Environments, an Unclassified Approach", 10th Interservice/Industry Training Systems Conference, pp. 151-160, November 1988.

AIAA Flight Simulation Technologies Conference and Exhibit
September 17-19, 1990 / Dayton, OH

AUTHOR INDEX

(Number after name indicates session in which paper appears)

A

Arndt, C. -2

B

Baarspul, M. -7
Bacha, J. -12
Baltrus, D. -1
Banks, S. -11
Batson, V. -2
Bromley, M. -13
Brown, P. -10
Brown, R. -13
Brown, Y. -5
Brumback, L. -12
Bunnell, J. -8
Burnett, J. -6
Burnett, W. -12
Buttrill, C. -7

C

Cadiz, J. -6
Calspan, V. -8
Cardullo, F. -5
Carico, D. -4
Clark, W. -11
Clayton, G. -13
Cleveland, J. -1
Crawford, D. -1
Cress, J. -8

D

Davoudian, A. -3
Day, C. -8
Draffin, D. -12
Dutton, K. -4

E

Eccles, D. -13
Emerson, L. -11

F

Farley, J. -11
Fineberg, M. -9
Frey, F. -4

G

Galloway, R. -4.
George, G. -6
Goddard, D. -11
Goldiez, B. -10
Green, J. -12
Gupton, L. -2

H

Hackett, J. -11
Hajare, A. -10
Hampson, B. -7
Hayashigawa, L. -5
Hettinger, L. -5
Hicks, B. -3 -3
Hollstein, R. -1
Houck, J. -7
Howe, R. -9

J

Joglekar, M. -12
Johnson, W. -8

K

Katz, A. -6
Kawahara, H. -2
Knight, S. -6
Kurihara, Y. -2

L

Lester, L. -10
Levi, R. -5
Lickenbrock, J. -13
Lin, K. -9
Loper, M. -6
Lufkin, B. -13
Lusk, S. -8

M

Matsumura, K. -2.
Matusof, R. -3
McBride, D. -10
McCauley, M. -5
McFarland, R. -8
McMillan, G. -5, 8
Middendorf, M. -8
Moerder, D. -4
Montag, B. -4, 11
Moore, R. -11
Moriarty, P. -1
Morik, F. -2
Moxon, B. -12
Mulder, J. -7

N

Nahon, M. -5

O

Oda, K. -13
Ohyama, S. -2
Ouyang, R. -6

P

Parker, L. -9

Q

Queen, E. -4

R

Reid, L. -5
Reynolds, P. -8
Roberts, B. -12
Robertson, J. -12

S

Sarnicola, J. -7.
Schwain, S. -3
Shimizu, T. -2
Sims, J. -10
Sinacori, J. -5
Sinacori, J. B. -5
Spuhl, K. -13
Stark, G. -7
Sterling, M. -10
Sudik, S. -1
Suit, W. -4
Swaine, S. -6

T

Takeda, K. -2
Thompson, J. -6

U

Udagawa, T. -2
Ure, R. -11

V

Vicroy, D. -4
Vliano, K. -10

W

Wagner, W. -3
Wakairo, K. -2
Walker, R. -5
Warden, G. -1
Watanabe, A. -2
Whiteley, J. -8
Wilsey, C. -1
Wilson, C. -5
Wurtz, J. -1

Z

Zical, W. -9

